(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) Übersetzung der europäischen Patentschrift

(51) Int. Cl.[7]: G 06 F 9/44

(97) EP 0 616 707 B 1

(10) DE 691 31 603 T 2

(21) Deutsches Aktenzeichen: 691 31 603.1
(86) PCT-Aktenzeichen: PCT/US91/09205
(96) Europäisches Aktenzeichen: 92 906 036.6
(87) PCT-Veröffentlichungs-Nr.: WO 93/12482
(86) PCT-Anmeldetag: 9. 12. 1991
(87) Veröffentlichungstag der PCT-Anmeldung: 24. 6. 1993
(97) Erstveröffentlichung durch das EPA: 28. 9. 1994
(97) Veröffentlichungstag der Patenterteilung beim EPA: 8. 9. 1999
(47) Veröffentlichungstag im Patentblatt: 6. 4. 2000

(73) Patentinhaber:
Digital Equipment Corp., Maynard, Mass., US

(74) Vertreter:
Grünecker, Kinkeldey, Stockmair & Schwanhäusser, 80538 München

(84) Benannte Vertragstaaten:
DE, FR, GB, IT

(72) Erfinder:
KIRK, Steven, A., Chelmsford, MA 01824, US;
BARABASH, William, Acton, MA 01720, US;
YERAZUNIS, William, S., Marlboro, MA 01752, US

(54) EXPERTEN SYSTEM MIT SCHNELLEM PATTERN-MATCHING UNTER VERWENDUNG VON ÄQUIVALENZ-KLASSEN

DE 691 31 603 T 2

DE 691 31 603 T 2

28.09.99

EP.-Az.: 92 906 036.6-2201

DIGITAL EQUIPMENT CORPORATION

Hintergrund der Erfindung

Diese Erfindung betrifft Struktur-Anpassung, wie sie bei komplexen Datenbasis-Systemen, und besonders bei auf Regeln beruhenden Systemen künstlicher Intelligenz (d.h sogenannten "Expertensystemen"), entsteht, bei denen die durch die Regeln bestimmten Bedingungen als ein Netzwerk strukturiert sind, um ein wirksame Bestimmung zuzulassen, welche Regeln auf von dem System erhaltenen Daten anwendbar sind.

Ein Expertensystem beruht auf einem Computerprogramm, welches einen Satz von Regeln und Definitionen von Datenarten umfaßt, die gemäß den Regeln bearbeitet werden. Durch einen als "Inferenz" (logisches Schließen) bekannten Vorgang führt das System eine Anpassung oder Abgleichung zwischen Bedingungen aus, die in dem Regelsatz und in Datenelementen der definierten Arten in einer arbeitenden Datenbasis dargelegt sind, die durch das System von einem Benutzer oder von anderen Datenbasen erhalten wurden. Wenn ein Satz von Datenelementen die Bedingungen in einer oder mehrerer der Regeln erfüllt, werden eine oder mehr der erfüllten Regeln gefeuert (d.h. auf die Daten angewendet), um die Datenelemente in der arbeitenden Datenbasis zu ändern und Rat für den Benutzer oder Erklärungen in Abhängigkeit von den Fragen des Benutzers zu schaffen oder irgendeine andere Maßnahme zu ergreifen. Der Inferenzvorgang wird wiederholt, bis keine Regeln mehr als durch die Daten erfüllt gefunden werden, oder das Feuern einer Regel bezeichnet, daß der Inferenzvorgang anzuhalten ist.

Im allgemeinen kann der Inferenzvorgang als ein Versuch der Bewertung eines Satzes von Objekten (alle Datenelemente in der arbeitenden Datenbasis) im Lichte eines vorher vorhandenen Satzes von Constraints (Begrenzungen (die Bedingungen in der Regel) gedacht werden, um eine Vielzahl von Objekten zu finden, welche den Satz von Constraints erfüllen (die Kombination der Datenelemente, welche die Regel erfüllen).

Jede Regel enthält eine oder mehrere Bedingung(en) (kollektiv als die "linke Seite" der Regel bezeichnet), welche, wenn sie durch die Daten erfüllt werden, bezeichnen, daß die Regel anwendbar ist. Nachdem die Bedingungen der Regeln bewertet sind, um zu bestimmen, welche Regeln anwendbar sind, wird eine oder werden mehrere der anwendbaren Regeln gefeuert. Die nach dem Feuern der Regeln auszuführenden Vorgänge werden in der "rechten Seite" der Regel aufgelistet. Ein Weg, um zu bestimmen, welche Regeln anwendbar sind, besteht darin, jede Bedingung in den linken Seiten aller Regeln jedesmal zu prüfen, wenn ein neues Datenelement durch das System erhalten wird. Mit diesem Schema wird jede Bedingung jedesmal neu geprüft, wenn die arbeitende Datenbasis durch Hinzufügung oder Beseiti-

gung von Daten geändert wird, auch wenn die geänderten Daten das Ergebnis dieser Bedingung möglicherweise nicht geändert haben.

Es ist bekannt, daß während des Betriebs eines Expertensystems der größte Zeitanteil beim Bewerten der Bedingungen in den Regeln damit verbracht wird, zu bestimmen, welche Regeln anwendbar sind. Es ist deshalb nötig, die beim Bewerten der Bedingungen in den Regeln verbrauchte Zeit zu minimieren, um die Verhaltensweise von Expertensystemen zu verbessern,.

Eine Vorgehensweise aufgrund satzorientierter Konstrukte wird gegeben in dem Aufsatz von D.N. Gordin u.a.: "Set oriented constructs for rule-based Systems", 7th Proc. on the IEEE Conf. on AI, Band, 19, Seiten 76-80, Los Alamitos, CA, Februar 91. Diese Konstrukte erlauben es, willkürliche Größen von Daten innerhalb der Ausführung einer einzelnen Regel anzupassen und zu ändern.

Eine weitere Vorgehensweise, die zum Minimieren von Bedingungsbewertungszeit vorgenommen wird, benutzt ein TREAT-Netz, das von Daniel Miranker in TREAT: A New and Efficient Match Algorithm for AI Production Systems (Research Note in Artificial Intelligence - Morgan Kaufmann Publishers, Inc.) vollständig beschrieben ist. Ein TREAT-Netz ist eine Datenstruktur, durch welche die Bedingungen aller Regeln in einem Netzwerk von Knoten bewertet werden. Die Bedingungen der Regeln werden in ausgewählten Knoten (und ihren zugehörigen Speichern) geprüft, die durch einen Satz von Verbindungsknoten miteinander verbunden sind, um die verschiedenen in der linken Seite der Regeln angegebenen Kombinationen von Bedingungen darzustellen. Zeichen, welche Datenelemente oder Sätze von Datenelementen in der Arbeitsdatenbasis repräsentieren, werden über die Verbindungen der Knoten durch das TREAT-Netz hindurchgeleitet, wenn die Datenelemente die verschiedenen Prüfungen und Prüfungskombinationen durchlaufen. Das TREAT-Netz schließt mit den jeweiligen Einzelregeln entsprechenden Ergebnisknoten ab. Wenn die linke Seite einer Regel erfüllt ist, speichert das TREAT-Netz ein Zeichen, welches den Satz von Datenelementen repräsentiert, der die Bedingung der Regel in dem dieser Regel entsprechenden Ergebnisknoten erfüllt. Der Satz von Zeichen, der dem Ergebnisknoten zugeführt wird, ist als ein Konfliktsatz bekannt, da er Zeichen enthält, welche alle möglichen Kombinationen von die Bedingungen erfüllenden Datenelementen repräsentieren.

Noch ein andere Vorgehensweise, die zum Minimieren der Bedingungsbewertungszeit in manchen Expertensystemen unternommen wurde, war, die linken Seiten der Regeln als ein "Entscheidungsnetz" zu strukturieren, wie ein sog. "retikulares" Netz oder RETE-Netz. Ein RETE-Netz bestimmt wirksam, welche Regeln anwendbar sind, indem, wenn die Arbeits-

datenbasis geändert wird, nur die Bedingungen überprüft werden, deren Ergebnisse sich in Abhängigkeit von den geänderten Daten in der Arbeits- Datenbasis geändert haben können.

Ein RETE-Netz, das durch Charles L. Forgy und Susan J. Shepard in RETE: A fast match algorithm (AI Expert, Januar 1987) vollständig beschrieben worden ist, besteht aus miteinander verbundenen Knoten einschließlich einem Satz von 1-Eingangs-Knoten (einschließlich zugehöriger Speicher) und einem Satz von 2-Eingangs-Knoten (ebenfalls einschließlich zugehöriger Speicher). Daten können als durch das RETE-Netz über die Verbindungen strömend gedacht werden. Jeder Knoten empfängt Datenelemente oder Sätze von Datenelemente repräsentierenden Zeichen (Token) von der Arbeits-Datenbasis oder von einem anderen Knoten, verarbeitet die empfangenen Daten und erzeugt möglicherweise andere Zeichen, welche durch die Verbindungen zu einem anderen Knoten weiterzuleitende Daten repräsentieren. Die 1-Eingangs-Knoten bewerten Bedingungen, die sich auf ein Einzeldatenelement beziehen, und die 2-Eingangs-Knoten bewerten Bedingungen, welche die Beziehung zwischen unterschiedlichen Datenelementen oder Sätzen unterschiedlicher Datenelemente betreffen. Das RETE-Netz schließt mit Ergebnisknoten ab, welche jeweiligen einzelnen Regeln entsprechen. Wenn die linke Seite einer Regel erfüllt ist, speichert das RETE-Netz in dem dieser Regel entsprechenden Ergebnisknoten ein Zeichen, welches die Datenelemente repräsentiert, die die Bedingungen dieser Regel erfüllen. Die Zeichen in dem Ergebnisknoten sind als ein Konfliktsatz bekannt, da sie alle Zeichen enthalten, welche die die Bedingungen dieser Regel erfüllenden Datenelemente repräsentieren.

Der Betrieb eines Expertensystemes umfaßt zwei Schritte, die Kompilierung (Umwandlung) und Exekution (Ausführung). Vor der Kompilierung erzeugt ein Programmierer ein Expertensystem-Quellenprogramm, welches die Regeln und Datendefinition des Expertensystems definiert. Dieses Quellenprogramm wird durch einen Kompilierer verarbeitet, um ein ausführbares Programm und zugeordnete Datenstrukturen zu erzeugen, die dann zum Verwirklichen der Vorgänge des Expertensystems ausgeführt werden können. Die während der Kompilierung erzeugten Datenstrukturen enthalten ein RETE-Netz oder TREAT-Netz zum Bewerten der linken Seiten der Regeln.

Vor der Ausführung werden das kompilierte ausführbare Programm und die zugeordneten Datenstrukturen einschließlich des RETE-Netzes oder TREAT-Netzes zum Bewerten der linken Seiten der Regeln in einen Speicher eines Computersystems geladen. Zusätzlich wird in Abschnitt dieses Speichers einer Arbeitsdatenbasis zugewiesen, um aktuelle durch das Expertensystem zu verarbeitende Datenelemente zu enthalten. Dieses Computersystem führt dann den Inferenzvorgang aus und ist als Inferenzmaschine bekannt. Das ausführbare Pro-

gramm läßt man dann durch die Inferenzmaschine laufen, um die aktuellen Datenelemente in der Arbeitsdatenbasis durch das RETE-Netz oder TREAT-Netz zu verarbeiten und die Expertensystemfunktionen aktuell auszuführen.

Fig. 1 ist ein Blockschaltbild eines bekannten Computersystems, das ein Expertensystem mit einem RETE-Netz oder TREAT-Netz verkörpert. In Fig. 1 enthält das Expertensystem 10 ein Expertensystem-Quellenprogramm 12 und einen Kompilierer 20, der das Expertensystem-Quellenprogramm 12 zum Erzeugen eines ausführbaren (nicht gezeigten) Programms und zugehöriger Datenstrukturen (die mit mehr Einzelheiten später beschrieben werden). Das Expertensystem 10 kann mit Benutzung jeder geeigneten Expertensystem- Programmierungssprache (wie OPS5) geschaffen werden. Das Expertensystem-Quellenprogramm 12 enthält einen Satz von programmversorgten Datendefinitionen 14, welche Klassen von Daten identifizieren, die das Expertensystem 10 während der Ausführung verwendet. Während der Ausführung wird der Speicher zum Speichern aktueller Datenelemente, die durch die Datendefinitionen 14 definiert sind, einer Arbeitsdatenbasis zugeordnet. Wenn z.B. das Expertensystem-Quellenprogramm 12 ein Expertensystem für das Abgeben von das Anziehen von Leuten betreffenden Ratschlägen definiert, können die Datendefinitionen 14 Klassen wie HEMDEN, HOSEN, KRAWATTEN und SOCKEN enthalten.

Das Expertensystem-Quellenprogramm 12 enthält auch einen Satz von vom Programmierer zugeführten Regeln, die durch die Regeldefinitionen 16 identifiziert sind, welche die Art beschreiben, mit welcher aktuelle Daten, welche die Datendefinitionen 14 erfüllen, durch das Expertensystem 10 während der Ausführung manipuliert werden, und wie das Expertensystem 10 mit dem Benutzer und dem Rest des Computersystems zusammenwirken wird.

Das Expertensystem-Quellenprogramm 12 kann auch programmdefinierte Verbindungen zu anderen Quellenprogrammen 18 in anderen Sprachen (wie C, FORTRAN, BASIC usw.) enthalten, um Aufgaben wie den Zugriff zu entfernt liegenden Datenbasen oder das Herstellen von statistischen Berechnungen auszuführen, wenn solche während der Ausführung des Expertensystems 10 aufgerufen werden. Diese anderen Quellenprogramme 18 in anderen Sprachen werden durch jeweilige Kompilierer 22 für diese anderen Sprachen kompiliert, um jeweilige ausführbare Programme und zugehörige Datenstrukturen zu erzeugen.

Die vom Programmierer zugeführte Datendefinitionen 14 und Regeldefinitionen 16 werden durch den Kompilierer 20 kompiliert und das sich ergebende Programm und die zugehörigen Datenstrukturen in einen (nicht gezeigten) Abschnitt des Speichers 30 in einer Inferenzmaschine 24 gespeichert. Insbesondere ist der Abschnitt 26 des Speichers 30 dem RETE-Netz oder TREAT-Netz zugeordnet, welches die linke Seit der Regeldefinition 16 dar-

stellt, und Abschnitt 34 des Speichers 30 ist der Arbeitsdatenbasis zugeordnet, welche ein Vielzahl von aktuellen Datenelementen 36 speichern kann. Die anderen Programme 18 werden durch ihre eigenen Kompilierer 22 kompiliert und die ausführbaren Programme und zugehörigen Datenstrukturen in anderen (nicht gezeigten) Abschnitten des Speichers 30 gespeichert. Ein Prozessor und eine Steuerung 28 in der Inferenzmaschine 24 steuert den Betrieb der Kompilierer 20 und 22 während des Kompilierens. Der Prozessor und die Steuerung 28 führen auch durch den Kompilierer 20 erzeugte ausführbare Expertensystem-Programme aus und empfangen Daten von und senden Daten zu einem Benutzer während der Ausführung durch eine Benutzerschnittstelle 32.

Fig. 2 ist ein Textschaubild, das eine bekannte OPS5-Quellendarstellung eines Abschnittes einer Regel 60 darstellt, die in dem in Fig. 1 dargestellten Expertensystem benutzt werden kann. Für die Regel 60 in Fig. 2 haben vorher verarbeitete (nicht gezeigte) Datendefinitionen in bekannter Weise Datenklassen definiert: HEMDEN, HOSEN, KRAWATTEN und SOCKEN. Ebenfalls in bekannter Weise können allen Datenklassen (durch das Prefix "^") bezeichnete ihnen zugeordnete Attribute besitzen. In Fig. 2 sind allen Datenklassen das Attribut ^FARBE zugeordnet. Eine Klasse zugeordnete Attribute werden als ein Wert bezeichnet, der durch die Bedingungen der linken Seite der Regel in dem Expertensystem geprüft werden kann.

In Regel 60 besteht die linke Seite 65 aus Bedingungen in Zeilen 70, 72, 74 und 76. Zeile 70 bezeichnet, daß ein Datenelement der HEMDEN-Klasse in der Arbeitsdatenbasis 34 (aus Fig. 1) vorhanden sein muß, damit die Regel erfüllt ist. Zusätzlich wird der Wert des ^FARBE-Attributs des HEMDEN-Klassendatenelements in einer Variablen <COL> gespeichert (bezeichnet durch zwei spitze Klammern <...>, welche den Namen der Variablen umschließen). Zeilen 72 und 76 bezeichnen, daß auch jeweils Datenelemente der HOSEN-Klasse, KRAWATTEN-Klasse und SOCKEN-Klasse vorhanden sein müssen und, gemäß der OP5-Sprache der Wert der ^FARBE-Attribute der Datenelemente dieser Klassen der gleiche wie der vorher in der <COL>-Variablen gespeicherte sein muß. (D.h. das ^FARBE-Attribut der Datenelemente der HOSEN-Klasse, KRAWATTEN-Klasse und SOKEN-Klasse muß jeweils den gleichen Wert wie das ^FARBE-Attribut des HEMDEN-Klassen-Datenelementes besitzen.)

Die Regel 60 ist nur dann vollständig erfüllt, wenn die Daten allen Tests 70, 72, 74 und 76 genügen. Wenn die Regel 60 vollständig erfüllt ist, wird ein Zeichen (Token), das Verweise zu den vier Datenelementen, welche die Regel 60 erfüllen, in einem der Regel 60 zugeordneten Ergebnisknoten gespeichert.

Nach Fig. 2 kann die Variable <COL>, wie sie in Zeile 70 verwendet wird, von der Variablen <COL>, wie sie in Zeilen 72 bis 74 benutzt wird, differenziert werden. Die in Zeile 70

verwendete Variable <COL> ist als eine 'gebundene' Variable bekannt. Wenn eine OP5-Variable das erste Mal in einer Regel benutzt wird, wirkt die Variable als Wildcard, die sich anpaßt und der jeder Wert zugeordnet wird, der sich in dem Attribut des Datenelementes von der Arbeitsdatenbasis befindet, die gegenwärtig geprüft wird. Das erste Mal, bei dem eine Variable dem Wert eines Attributs eines Datenelementes in der Arbeitsdatenbasis angepaßt wird, wird gesagt, daß der Wert dieses Attributs dieser Variablen verbunden ist. Jede darauffolgende Verwendung der gleichen Variablen in dieser Route stellt den vorher gebundenen Wert dar. Als ein Beispiel, wenn der Wert des ^FARBE-Attributs des HEMDEN-Klassen-Datenelementes blau ist, dann wird der Variablen <COL>, wie sie in70 verwendet wird, der Wert blau zugeordnet. Wenn die Variable <COL> wieder bei der Regel 60 benutzt wird, entweder in der linken Seite 65 (wie in den Zeilen 72 bis 76) oder der rechten Seite 78, stellt sie den Wert blau dar.

Die Variable <COL>, wie sie in Zeilen 72 bis 76 benutzt wird, ist als eine 'geprüfte' Variable bekannt. Eine geprüfte Variable ist eine solche, auf die in einer Regel vorher Bezug genommen wurde und der so ein Wert zugeordnet war. Der Wert des bezeichneten Attributs des Datenelementes, das gegenwärtig geprüft wird, wird mit dem vorher dieser Variablen (bei dem ersten Auftreten dieser Variablen) zugeordneten Wert verglichen und muß zur Konsistenz ihm gleich sein. In Fortsetzung des vorigen Beispiels müssen, wenn die Variable <COL> in Zeile 70 einen Wert blau zugeordnet hatte, in den Zeilen 72 bis 76 die jeweiligen Werte der ^FARBE-Attribute der HOSEN-Klasse, KRAWATTEN-Klasse und SOCKEN-Klasse-Datenelementen alle mit dem Wert der Variablen <COL> verglichen werden und müssen zur Konsistenz ihr gleich sein. Das bedeutet, nur, wenn die Werte der ^FARBE-Attribute der HOSEN-Klasse, KRAWATTEN-Klasse und SOCKEN-Klasse Datenelemente alle blau sind, sind die Bedingungen der Zeilen 72 bis 76 erfüllt.

Die rechte Seite 78 der Regel 60 enthält eine "Aktionsliste" 80 (im einzelnen nicht gezeigt), die ausgeführt wird, wenn die Regel gefeuert wird. Z.B. kann das Feuern der Regel herbeiführen, daß das System 10 in Fig. 1 mehr Daten ableitet oder den Wert einiger Daten berechnet (z.B. durch Ausführen eines der anderen Programme 18). Ein Regelfeuern kann auch das Aufnehmen einer externen Aktion veranlassen (wie das Anzeigen von Daten oder Ändern einer externen Datenbasis 38).

Fig. 3 ist ein Datenflußdiagramm eines bekannte TREAT-Netzes 190 entsprechend den Bedingungen der Regel 60 der Fig. 2. In Fig. 3 liefert ein Knoten 100 einer Arbeitsdatenbasis WDB Zeichen, welche alle Datenelemente 36 in der Arbeitsdatenbasis 34 (aus Fig. 1) repräsentieren, zu dem TREAT-Netz 190. Diese Zeichen werden durch das TREAT-Netz 190 unge-

ändert zu einem Terminal-Knoten 140 durchgelassen. Der Terminal-Knoten 140 kann mit darauffolgenden (nicht gezeigten) Wahlknoten gekoppelt sein, die anderen (ebenfalls nicht gezeigten) Regeln zugeordnet sind. Das TREAT-Netz 190 bewertet die Zeichen vom WDB-Knoten 100 und liefert einen Satz von Zeichen, welche Referenzen zu Datenelementen enthalten, welche die Regel 60 erfüllen, zu einem Ergebnisknoten 130 für die Regel 60. Die dem Ergebnisknoten 130 zugelieferten Zeichen repräsentieren den Konfliksatz für Regel 60.

Im allgemeinen werden die Bedingungen der Zeilen 70 bis 76 der Regel 60 durch das TREAT-Netz 190 in einer bekannten Weise bewertet. In Fig. 3 bewerten die Wahlknoten 110a - 110d die Anteile der jeweiligen Bedingungen an Zeilen 70 - 76, die sich nur auf diese Bedingung beziehen. Die Verbindungsknoten 120a - 120d bewerten die Anteile der Bedingungen an Zeilen 70 - 76, die sich auf jeweils unterschiedliche Bedingungen beziehen. Das Ergebnis der in den Verbindungsknoten 120a - 120d ausgeführten Untersuchungen werden dem Ergebnisknoten 130 zugeführt.

Jeder Wahlknoten (110a - 110d) hat einen einzelnen Eingabeanschluß (allgemein an der linken Seite des Knotens dargestellt), bei dem Datenelemente von der Arbeitsdatenbasis darstellende Zeichen aufgenommen werden. Jeder Wahlknoten (110a - 110d) enthält auch einen Durchgangsanschluß (allgemein an der rechten Seite des Knotens gezeigt), von welchem alle dem Eingabeanschluß des Knotens zugeleiteten Zeichen ungeändert zu darauffolgenden Knoten in dem TREAT-Netz 190 durchgeleitet werden. Jeder Wahlknoten (110a - 110d) besitzt auch einen (allgemein an der Unterseite des Knotens gezeigten) Ausgabeanschluß, von welchem Zeichen, die Datenelemente darstellen, welche den bestimmten in dem Knoten ausgeführten Test erfüllt haben, zu darauffolgenden Knoten in dem TREAT-Netz 190 durchgeleitet werden (wie mit mehr Einzelheiten später beschrieben wird). Schließlich besitzt jeder Wahlknoten 110a - 110d einen jeweiligen zugeordneten internen Speicher 111a - 111d, in welchem Kopien von zum Ausgabeanschluß zugelieferten Zeichen gespeichert sind.

Jeder Verbindungsknoten 120a - 120d hat einen Eingabeanschluß, der allgemein an der Unterseite des Knotens gezeigt ist, an welchem Zeichen von den vorhergehenden Wahlknoten empfangen werden. Jeder Verbindungsknoten enthält auch Verbindungsanschlüsse, die allgemein an den Seiten der Knoten gezeigt sind, zum Absenden und Empfangen von Datenpaketen, welche Sätze von teilweise angepaßten Datenzeichen von einem Verbindungsknoten zu einem anderen repräsentieren. Zur Vereinfachung sind in Fig. 3 Zwischenverbindungsanschlüsse nur für einander benachbarte Verbindungsknoten dargestellt. Es ist jedoch zu verstehen, daß irgendein Verbindungsknoten solche Datenpakete zu jedem anderen senden oder von ihm empfangen kann. Jeder Verbindungsknoten 120a - 120d hat auch einen Ausga-

beanschluß, der allgemein an der Unterseite des Knotens gezeigt ist, von dem Zeichen zu dem Ergebnisknoten 130 geliefert werden, welche Kombinationen von die Regel erfüllend n Datenelementen darstellen. Zusätzlich ist jeder Verbindungsknoten 120a - 120d mit allen örtlichen Speichern 111a - 111d aller Verbindungsknoten 110a - 110d über Anfragedatenpfade verbunden. Es sind jedoch nur die Anfragedatenpfade 121a - 121d für Verbindungsknoten 120a dargestellt, um die Fig. 3 zu vereinfachen.

Besonders enthält das TREAT-Netz 190 Wahlknoten 110a - 110d mit seriell gekoppelten Eingangs- und Durchleitanschlüssen, die zwischen dem WDB-Knoten 100 und dem Terminal-Knoten 140 gekoppelt sind. Die Ausgabeanschlüsse der Wahlknoten 110a - 110d sind mit jeweiligen Eingabeanschlüssen von Verbindungsknoten 120a - 120d gekoppelt. Die Verbindungsknoten 120a - 120d sind alle über ihre jeweiligen Verbindungsanschlüsse miteinander verbunden. Die Ausgabeanschlüsse aller Verbindungsknoten 120a - 120d sind mit dem Ergebnisknoten 130 gekoppelt. Der Verbindungsknoten 120a besitzt jeweilige Anfragedatenpfade 121a - 121d, die mit allen internen Speichern 111a - 111d aller Wahlknoten 110a - 110d gekoppelt sind. Die Verbindungsknoten 120b - 120d besitzen in gleicher Weise (nicht gezeigte) Anfragedatenpfade, die mit allen internen Speichern 111a - 111d aller Wahlknoten 110a - 110d gekoppelt sind.

Im Betrieb empfängt der Wahlknoten 110a Zeichen von dem WDB-Knoten 100 an seinem Eingangsanschluß, welche die in der Arbeitsdatenbasis 34 (der Fig. 1) gespeicherten Datenelemente repräsentieren, und leitet sie ungeändert zu dem Durchleitanschluß durch. Der Wahlknoten 110a entspricht der Bedingung an Zeile 70 der Fig. 2 und bewertet das durch das jeweilige Zeichen repräsentierte Datenelement, um zu bestimmen, ob es von der HEMDEN-Klasse ist. Ist es von der HEMDEN-Klasse, wird dieses Zeichen in dem internen Speicher 111a des Wahlknotens 110a gespeichert und über seinen Ausgabeanschluß zu dem Verbindungsknoten 121a durchgeleitet. Die Wahlknoten 110b - 110d arbeiten in einer gleichartigen Weise zum Empfangen von Zeichen an ihren jeweiligen Eingangsanschlüssen, ungeändertem Durchleiten dieser Zeichen zu ihren jeweiligen Durchleitanschlüssen und Plazieren von Zeichen der HOSEN-Klasse, KRAWATTEN-Klasse und SOCKEN-Klasse jeweils in den zugehörigen internen Speichern (111b - 111d) und Durchleiten dieser Zeichen zu ihren jeweiligen Ausgabeanschlüssen.

In Kombination bauen die Verbindungsknoten 120a - 120d in bekannter Weise Zeichen auf, welche die Kombinationen von Datenelementen repräsentieren, die die Bedingungen der Regel 60 der Fig. 2 erfüllen. Wenn irgendeiner der Verbindungsknoten 120a - 120d von seinem entsprechenden Wahlknoten 110a - 110d ein ein Datenelement repräsentierendes Zei-

chen erhält, wird ein Datenpaket erzeugt. Das Datenpaket besteht aus einer Aufzeichnungsliste, von denen jede Aufzeichnung einen Satz von Datenelement-Identifizierern und eine Anzeige der in dieser Anzeige enthaltenen Zahl solcher Identifizierer enthält. Dieses Datenpaket wird von Verbindungsknoten zu Verbindungsknoten durchgeleitet, bis jeder Verbindungsknoten durchschritten wurde. Von den Anfrageleitungen 121a - 121d erhaltene Daten werden benutzt, um zu bestimmen, zu welchem der Verbindungsknoten 120a - 120d das Paket durchzuleiten ist. Z.B. ist es bekannt, die Anzahl von Einträgen in jedem der internen Speicher in den anderen Wahlknoten 110a - 110d zu erreichen, um zu bestimmen, welche Verbindungsknoten als nächstes das Datenpaket verarbeiten sollten.

Wenn jeder Verbindungsknoten das Datenpaket erhält, vergleicht er der Reihe nach die bezogenen Datenelemente in jeder Aufzeichnung in dem Datenpaket mit den in dem internen Speicher seines entsprechenden Wahlknotens angezogenen Datenelementen, um zu bestimmen, ob irgendwelche Kombinationen von Datenelementen die durch diesen Verbindungsknoten durchgeführten Prüfungen erfüllen. Wenn solche Kombinationen gefunden werden, wird für jeden eine Aufzeichnung in dem Datenpaket geschaffen. Die neue Aufzeichnung besteht aus den Datenelement-Identifizierern von der Aufzeichnung von dem gegenwärtig geprüften Datenpaket mit einem Identifizierer für das daran angehängte passende Datenelement von dem internen Speicher. Der Anzeiger der Anzahl von Datenelement-Identifizierern in dieser Aufzeichnung wird dann (um 1) aufgezählt.

Wenn alle Verbindungsknoten durchlaufen worden sind, stellen die Aufzeichnungen in dem Datenpaket die Kombinationen von Datenelementen dar, welche an die Regel angepaßt sind. Ein Zeichen wird erzeugt, welches jede Kombination repräsentiert, die durch jede Aufzeichnung in dem Datenpaket repräsentiert wird, und die erzeugten Zeichen werden über den Ausgabeanschluß des zuletzt durchlaufenen Verbindungsknotens zu dem Ergebnisknoten 130 durchgeleitet, wo sie ein Teil des Konfliktsatzes für Regel 60 werden.

Jedes Zeichen in dem Konfliktsatz repräsentiert einen Satz von Datenelementen, die in Kombination die Regel 60 erfüllen, und bezeichnet, daß Regel 60 nun zum Feuern durch Ausführen der Aktionsliste 80 in der rechten Seite 75 der Regel 60 verfügbar ist. Das Verfahren des Auswählens einer Regel zum Feuern und das Feuern dieser Regel wird im einzelnen nachstehend diskutiert.

Fig. 4 ist ein Datenfluß-Schaubild eines bekannten RETE-Netzes 290 entsprechend den Bedingungen der Regel 60 der Fig. 2. In Fig. 4 liefert ein Arbeitsdatenbasis-(WDB)Knoten 200 dem RETE-Netz 290 Zeichen zu, welche alle Datenelemente 36 in der Arbeitsdatenbasis 34 (der Fig. 1) repräsentieren. Diese Zeichen werden durch das RETE-Netz 290 ungeändert zu

einem Terminal-Knoten 240 durchgeleitet. Der Terminal-Knoten 240 kann durch darauffolgende (nicht gezeigte) RETE-Netzknoten gekoppelt sein, die anderen (ebenfalls nicht gezeigten) Regeln zugeordnet sind. Das RETE-Netz 290 bewertet die Zeichen von dem WDB-Knoten 200 und liefert einen Satz von Zeichen, die jeweilige Kombinationen von Datenelementen repräsentieren, welche Regel 60 erfüllen, einem Ergebnisknoten 230 für Regel 60 zu. Die durch den Ergebnisknoten 230 zugeführten Zeichen repräsentieren den Konfliktsatz für Regel 60.

Die Bedingungen der Regel 60 der Fig. 4 werden in bekannter Weise durch ein Netz von 1-Eingangs-Knoten 210a - 210d und 2-Eingangs-Knoten 220a - 220c repräsentiert. Der Kompilierer 20 (der Fig. 1) erzeugt die Struktur des RETE-Netzes 290 in einer bekannten Weise in Abhängigkeit von den Datendefinitionen 14 und Regeldefinitionen 16.

Jeder 1-Eingangs-Knoten (210a - 210d) besitzt einen einzelnen Eingangsanschluß (allgemein an der linken Seite des Knotens gezeigt), an welchem Zeichen empfangen werden, welche Datenelemente von der Arbeitsdatenbasis repräsentieren; einen (allgemein an der rechten Seite des Knotens gezeigten) Durchleitanschluß, von welchem alle zu dem Knoten an dem Eingangsanschluß gelieferten Zeichen ungeändert zu nachfolgenden Knoten in dem RETE-Netz 290 geliefert werden; und einen (allgemein an der Unterseite des Knotens gezeigten) Ausgabeanschluß, von welchem Zeichen, die den bestimmten in diesem Knoten ausgeführten Test erfüllende Datenelemente repräsentieren, zu darauffolgenden Knoten in dem RETE-Netz durchgeleitet werden (wie mit mehr Einzelheiten später beschrieben wird). Jeder 1-Eingangs-Knoten 210a - 210d besitzt auch einen ihm zugeordneten jeweiligen Speicher 211a - 211d.

Die in den 1-Eingangs-Knoten (210a - 210d) eingesetzten Prüfungen betreffen einzelne Datenelemente von der Arbeitsdatenbasis. Im Betrieb führt jeder 1-Eingangs-Knoten in bekannter Weise seine zugeordnete Prüfung an jedem Zeichen aus, das an seinem Eingangsanschluß empfangen wurde. Wenn die durch diese Zeichen repräsentierten Daten den in diesem Knoten eingesetzten Test erfüllen, wird das Zeichen in dem mit dem Knoten verbundenen Speicher gespeichert, und dieses Zeichen wird durch den Ausgabeanschluß zu nachfolgenden Knoten in dem RETE-Netz 180 weitergeleitet.

Jeder 2-Eingangs-Knoten (220a - 220c) hat einen ersten oder linken (allgemein an der oberen linken Seite des Knotens gezeigten) Eingangsanschluß und einen zweiten oder rechten (allgemein an der oberen rechten Seite des Knotens gezeigten) Eingangsanschluß. Der linke Eingangsanschluß empfängt Zeichen, die entweder Datenelemente von der Arbeitsdatenbasis repräsentieren, welche in den vorhergehenden 1-Eingangs-Knoten (wie im 2-

Eingangs-Knoten 120a) eingesetzte Prüfungen erfüllen, oder repräsentieren Sätze von Datenelementen, die die in vorhergehenden 2-Eingangs-Knoten (wie in den 2-Eingangs-Knoten 220b und 220c) eingesetzte Prüfungen erfüllen. Die rechten Eingangsanschlüsse empfangen Zeichen, welche Datenelemente von der Arbeitsdatenbasis repräsentieren, welche in vorhergehenden 1-Eingangs-Knoten eingesetzte Prüfungen erfüllen. Jeder 2-Eingangs-Knoten (220a - 220c) enthält weiter einen (allgemein an der Unterseite des Knotens gezeigten) Ausgangsanschluß, von welchem Zeichen, welche Sätze von den in diesem 2-Eingangs-Knoten (220a - 220c) eingesetzte Prüfungen erfüllenden Datenelementen repräsentieren, zu darauffolgenden Knoten in dem RETE-Netz 290 weitergeleitet werden. Jeder 2-Eingangs-Knoten 220a - 220c besitzt auch ihm jeweils zugeordnete linke Speicher 221a - 221c und 221e und rechte Speicher 221b, 221d und 221e.

Die in 2-Eingangs-Knoten (130a - 130c) eingesetzten Prüfungen beziehen sich auf die Beziehung zwischen den verschiedenen Datenelementen in der Arbeitsdatenbasis. Im allgemeinen Betrieb speichert jeder 2-Eingangs-Knoten in bekannter Weise jedes Zeichen, das von dem linken Eingangsanschluß empfangen wurde, auf der linken Seite und speichert jedes Zeichen, das von dem rechten Eingangsanschluß empfangen wurde, in dem rechten Speicher. Immer, wenn ein Zeichen von einem Eingangsanschluß erhalten wird, wird das neu empfangene Zeichen in dem entsprechenden Speicher gespeichert. Dann wird das Datenelement oder der Datenelementesatz, das bzw. der durch das neu empfangene Zeichen repräsentiert wird, mit den Datenelementen kombiniert, die durch jedes andere in dem anderen Speicher gespeicherte Zeichen repräsentiert werden. Ein Zeichen, das jede solche Kombination repräsentiert, wird dann zu dem Ausgangsanschluß dieses 2-Eingangs-Knotens geleitet. Im Allgemeinbetrieb wird deswegen ein 2-Eingangs-Knoten einen Satz von Zeichen erzeugen, der das Querprodukt genannt wird, und die jeweiligen Kombinationen des Datenelementes oder des Datenelementesatzes, die in dem jeweiligen dem linken Eingangsanschluß zugeführten Zeichen enthalten sind, mit den Datenelementen von jedem Zeichen, das dem rechten Eingangsanschlußzugeführt wurde, repräsentiert. Die Anzahl solcher Zeichen in diesem Satz ist gleich dem Produkt der Anzahl der an dem ersten Eingangsanschluß empfangenen Zeichen mal der Anzahl der an dem zweiten Eingangsanschluß empfangenen Zeichen.

Der 2-Eingangs-Knoten kann jedoch auch einen Vergleich zwischen den verschiedenen Datenelementen ausführen, die durch die neuerdings empfangenen und die vorher gespeicherten Zeichen dargestellt werden. In einem solchen Fall werden nur Zeichen, welche solche Kombinationen von Datenelementen repräsentieren, die die Bedingung erfüllen, auf welche in dem 2-Eingangs-Knoten geprüft wurde, in dem zu dem Ausgangsanschluß des 2-Eingangs-

Knotens durchgeleiteten Zeichen enthalten sein. Wenn beispielsweise eine vorher gebundene Variable in einem darauffolgenden Zustand geprüft wird, wird der Wert des Attributs des in den darauffolgenden Bedingungen angesprochenen Datenelementes mit dem Wert verglichen, der vorher in der Variablen in dem 2-Eingangs-Knoten gespeichert worden ist. Nur Zeichen, welche Kombinationen von Datenelementen repräsentieren, bei denen die Werte des Attributs und der Variablen gleich sind, werden durch den Ausgangsanschluß zu darauffolgenden Abschnitten des RETE-Netzes 290 geleitet.

Insbesondere wird das RETE-Netz 290 aus 1-Eingangs-Knoten 210a-d gebildet mit seriell verbundenen Eingangs- und Durchleitanschlüssen, die zwischen dem WDB-Knoten 200 und dem Terminal-Knoten 210 gekoppelt sind. Zeichen, welche alle Datenelemente in der Arbeitsdatenbasis repräsentieren, treten durch diesen Pfad hindurch. Die Ausgangsanschlüsse der 1-Eingangs-Knoten 210a und 210b sind durch den 2-Eingangs-Knoten 220a miteinander verbunden. Die Ausgangsanschlüsse des 2-Eingangs-Knotens 220a und des 1-Eingangs-Knotens 210c sind durch den 2-Eingangs-Knoten 220b miteinander verbunden. Die Ausgangsanschlüsse des 2-Eingangs-Knotens 220b und des 1-Eingangs-Knotens 210d sind durch den 2-Eingangs-Knoten 210 miteinander verbunden, und der Ausgangsanschluß des 2-Eingangs-Knotens 220c ist mit dem Ergebnisknoten 230 gekoppelt.

Im Betrieb sei beispielsweise angenommen, daß die Arbeitsdatenbasis Datenelemente der HEMDEN-Klasse (welche Hemden repräsentiert), Datenelemente der HOSEN-Klasse (welche Hosen repräsentiert), Datenelemente der KRAWATTEN-Klasse (welche Krawatten repräsentiert) und Datenelemente der SOCKEN-Klasse (welche Socken repräsentiert) enthält. Datenelemente von anderen Klassen (nicht gezeigt oder nachstehend besprochen) können ebenfalls in der Arbeitsdatenbasis vorhanden sein. Weiter sei angenommen, daß jedes Datenelement der HEMDEN-Klasse, HOSEN-Klasse, KRAWATTEN-Klasse und SOCKEN-Klasse ein FARBE-Attribut besitzt, mit einem Wert (der die Farbe des dargestellten Kleidungsstücks repräsentiert). Diese Datenelemente können auch andere Attribute haben.

Am 1-Eingangs-Knoten 210a werden alle Datenelemente in der Arbeitsdatenbasis repräsentierende Zeichen von dem Eingangsanschluß zu dem Durchlaßanschluß durchgeleitet, aber nur solche Zeichen, die Hemden repräsentieren, d.h. die Zeichen, die sich auf Datenelemente der HEMDEN-Klasse beziehen, werden in dem Knotenspeicher 211a gespeichert und zu dem Ausgangsanschluß zur Fortsetzung durch das RETE-Netz 290 hindurchgeleitet. In gleichartiger Weise werden an den 1-Eingangs-Knoten 210b - 210d nur Zeichen in den jeweiligen Knotenspeichern 211b - 211d gespeichert und weiter durch das RETE-Netz 290 befördert, die sich auf Datenelementen der HOSEN-Klasse, KRAWATTEN-Klasse bzw. SOCKEN-

Klasse beziehen.

Am 2-Eingangs-Knoten 220a werden an dem linken Eingangsanschluß die Zeichen von dem 1-Eingangs-Knoten 210a empfangen, welche Bezüge auf die HEMDEN-Klassen-Datenelemente enthalten. Diese Zeichen werden in dem linken Speicher 221a gespeichert. Bei jedem erhaltenen Zeichen ist der Wert des ^FARBE-Attributs des in diesem Zeichen enthaltenen HEMDEN-Klassen-Datenelements an die Variable <COL> gebunden. Dann wird jedes in dem rechten Speicher 221b gespeicherte Zeichen, das HOSEN-Klassen-Datenelemente repräsentiert, in der folgenden Weise bewertet. Der Wert des ^FARBE-Attributs des in dem vorher in dem rechten Speicher 221b gespeicherten Zeichen enthaltenen Datenelements wird dann mit dem Wert in der <COL> Variablen verglichen. Wenn die Werte gleich sind, wird ein sich auf die Kombination der angepaßten beiden Datenelemente beziehendes Zeichen erzeugt, und dieses Zeichen wird an den Ausgangsanschluß des 2-Eingangs-Knotens 220a weitergeleitet.

Die Zeichen von den 1-Eingangs-Knoten 210b, welche Bezüge auf die HOSEN-Klassen-Datenelemente enthalten, werden an dem rechten Eingangsanschluß empfangen. Diese Zeichen werden in dem rechten Speicher 221b gespeichert. Beim Erhalt jedes Zeichens wird es mit allen vorher in dem linken Speicher 221a gespeicherten Zeichen in der folgenden Weise verglichen. Der Wert des ^FARBE-Attributs der in jedem solchen vorher gespeicherten Zeichen enthaltenen HEMDEN-Klassen-Datenelemente ist an die Variable <COL> gebunden. Dann wird der Wert des ^FARBE-Attributs des in dem neuerdings von dem rechten Eingangsanschluß erhaltenen Zeichen enthaltenen HOSEN-Klassen-Datenelements mit dem Wert in der <COL> Variablen verglichen. Wenn die Werte gleich sind, wird ein sich auf die Kombination der angepaßten beiden Datenelemente beziehendes Zeichen erzeugt, und dieses Zeichen zu dem Ausgangsanschluß des 2-Eingangs-Knotens 220a weitergeleitet.

Die 2-Eingangs-Knoten 220b und 220c arbeiten in einer gleichartigen Weise: mit Bindung der Variablen <COL> an den Wert des ^FARBE-Attributs des in jedem Zeichen, ob es nun neuerdings von dem linken Eingangsanschluß empfangen wurde oder vorher in den linken Speicher 221c oder 221e gespeichert war, enthaltenen HEMDEN-Klassen-Datenelements, und Vergleichen des Werts der <COL> Variablen mit dem Wert des ^FARBE-Attributs des KRAWATTEN-Klassen- bzw. SOCKEN-Klassen Datenelements, das jeweils in dem Zeichen enthalten ist, das entweder vorher indem rechten Speicher 221d bzw. 221f gespeichert oder neuerdings von dem rechten Eingangsanschluß erhalten wurde. Wenn die Werte gleich sind, wird ein sich auf die Kombination der passenden Datenelemente beziehendes Zeichen erzeugt, und dieses Zeichen wird zu dem Ausgangsanschluß der 2-Eingangs-Knoten 220b und

220c durchgeleitet.

In jedem Fall nimmt der Zeitraum, der durch das TREAT-Netz oder das RETE-Netz beim Bewerten der Datenelemente in der Arbeitsdatenbasis verbracht wird, typischerweise den größten Teil der Ausführungszeit des Expertensystems ein. Es ist erwünscht, daß die Zeit, die bei der Bewertung der Datenelemente in der Arbeitsdatenbasis verbracht wird, minimiert wird, um den Betrieb eines solchen Expertensystems zu verbessern.

Gemäß der vorliegenden Erfindung wird ein Expertensystem nach dem kennzeichnenden Abschnitt des Anspruchs 1 geschaffen.

Die Erfinder haben realisiert, daß bei den TREAT-Netz- oder RETE-Netz-Verwirklichungen von auf Regeln beruhenden Expertensystemen nach dem Stand der Technik zeitweise viele Zeichen durch das Netz geleitet werden, welche mit Bezug auf die gerade bearbeitete Regel ununterscheidbar sind. Da diese Zeichen nicht unterschieden werden können, ist die Verarbeitung aller dieser Zeichen durch das TREAT-Netz oder RETE-Netz entsprechend dieser Regeln unnötig und ergibt eine unnötigerweise bei der Bedingungsbewertung verbrachte Zeit. Falls Zeichen, welche Datenelemente repräsentieren, die mit Bezug auf die gerade bewertete Regel nicht unterscheidbar sind, in eine Klasse eingereiht werden, die eine Äquivalenzklasse genannt wird und ein einziges diese Äquivalenzklasse repräsentierendes Zeichen durch das TREAT-Netz oder RETE-Netz hindurchgeleitet wird (statt alle getrennten Zeichen durchzuleiten), kann die Verarbeitungszeit minimiert und das Verhalten des Expertensystems optimiert werden.

Bei dem angeführten Beispiel kann die Arbeitsdatenbasis viele Zeichen enthalten, die HEMDEN-Klassen-Datenelemente mit dem gleichen Wert für das ^FARBE-Attribut besitzen. Das kann davon kommen, weil andere (nicht gezeigte) Attribute unterschiedliche Werte einnehmen (ein blaues Anzughemd und ein blaues Flanellhemd und ein blaues Polohemd usw.), oder einfach, weil diese Zeichen verschiedene unterschiedliche Gegenstände mit den gleichen Attributen repräsentieren (d.h. 10 identische blaue Hemden). Mit Bezug auf Regel 60 der Fig. 2 werden diese unterschiedlichen Zeichen alle in den Systemen nach dem Stand der Technik verarbeitet, jedoch sind die Ergebnisse der Verarbeitung identisch.

Genauer gesagt, bei dem angeführten Beispiel beziehen sich die vorstehend angezogenen bezeichnenden zwischen den Objekten liegenden Constraints (Begrenzungsreiationen) auf die Verwendung einer gebundenen und/oder getesteten Variablen in einer Regel. Der Kompilierer 20 der Fig. 1 wird derartige bezeichnende zwischen den Gegenständen liegende Constraints erkennen. In Regel 60 der Fig. 2 betrifft das bezeichnende Interobjekt-Constraint die Variable <COL>, die mit den ^FARBE-Attributen aller Datenelemente der HEMDEN-

Klasse, HOSEN-Klasse, KRAWATTEN-Klasse und SOCKEN-Klasse verglichen wird. Die erzeugten Äquivalenzklassen sind: alle HEMDEN-Klassen-Datenelemente mit dem gleichen Wert im ^FARBE-Attribut, alle HOSEN-Klassen-Datenelemente mit dem gleichen Wert im ^FARBE-Attribut, alle KRAWATTEN-Klassen-Datenelemente mit dem gleichen Wert im ^FARBE-Attribut und alle SOCKEN-Klassen-Datenelemente mit dem gleichen Wert im ^FARBE-Attribut. Dann werden einzelneZeichen, welche derartige Äquivalenzklassen repräsentieren durch das TREAT-Netz oder RETE-Netz verarbeitet. Wenn schließlich eine Kombination von Äquivalenzklassen ausgewählt wird, welche das Feuern der Regel herbeiführt, werden die Mitglieder jeder Äquivalenzklasse in dem ausgewählten Zeichen in dem Konfliktsatz bewertet, und ein Mitglied von jeder wird ausgewählt, um die Kombination von Datenelementen zu bilden, die als die Regel zum Feuern veranlaßt habend angesehen wird. Das wird das Zurück-Auftragen der Äquivalenzklassen zu den Datenelementen genannt.

Zusätzlich besteht bei einem auf Regeln beruhenden Expertensystem im besonderen keine Notwendigkeit, eine vollständige Liste von Kombinationen von die Regeln erfüllenden Datenelementen aufrecht zu erhalten. Stattdessen braucht nur der Einzelsatz, welcher die (in einer bekannten Weise bestimmte) "beste" Kombination repräsentiert, für jede Kombination von Äquivalenzklassen in dem Konfliktsatz aufrechterhalten zu werden. Das stellt eine Einsparung in der Speichergröße dar, die zum Aufrechterhalten des Konfliktsatzes benötigt wird.

Kurze Beschreibung der Zeichnungen

Die Verbesserungen der gegenwärtigen Erfindung gegenüber dem Stand der Technik und die sich daraus ergebenden Vorteile werden ersichtlicher beim Lesen der nachfolgenden Beschreibung der bevorzugten Ausführungsformen, mit Bezug auf die Zeichnungen, in welchen zeigt:

Fig. 1 ein funktionelles Blockschaltbild eines Computersystems, das ein Expertensystem verwirklicht;

Fig. 2 ein Textschaubild, das für eine in dem System der Fig. 1 benutzte Regel repräsentativ ist;

Fig. 3 ein Datenflußdiagramm einer TREAT-Netz-Repräsentation der Regel der Fig. 2;

Fig. 4 ein Datenflußdiagramm einer RETE-Netz-Repräsentation der Regel der Fig. 2;

Fig. 5 eine schematische Darstellung des Verfahrens gemäß der vorliegenden Erfindung;

Fig. 6 ein Datenflußdiagramm einer TREAT-Netz-Repräsentation der Regel nach Fig. 2 gemß der vorliegenden Erfindung, das weiter die Inhalte von ausgewählten Knoten in dem TREAT-Netz zugeordneten Speichern darstellt; und

Fig. 7 ein Schaubild einer RETE-Netz-Repräsentation der Regel nach Fig. 2 gemäß der

vorliegenden Erfindung, die weiter die Inhalte von ausgewählten Knoten in dem RETE-Netz zugeordneten Speichern darstellt.

Beschreibung der bevorzugten Ausführungsformen

Fig. 5 ist eine schematische Darstellung eines Verfahrens zum Finden eines Satzes von Tupeln von Objekten aus einem allgemeinen Satz dieser gleichen Objekte, welche einen bestimmten Satz von Constraints gemäß der vorliegenden Erfindung erfüllen. Das Verfahren beginnt mit einem bestimmten Satz 2 von Objekten. Bezeichnende Interobjekt-Constraints 4 werden dann aus einem bestimmten Constraint-Satz (wie er mit mehr Einzelheiten später beschrieben wird) identifiziert. Ein Satz von Äquivalenzklassen 6 aus dem Satz 2 von Objekten wird aufgrund der bezeichnenden Zwischenobjekt-Constraints 4 erzeugt. Eine Äquivalenzklasse ist ein Satz von Objekten, die für den Zweck einer komplexen Datenbasisuntersuchung (wie Einigungen über ein Nicht-Schlüsselfeld) in einem Datenbasissystem oder das Anpassen bei einer bestimmten Regel in einem auf Regel beruhenden Expertensystem, sind nicht unterscheidbar und vollständig austauschbar mit Bezug auf die ausgeführte Datenbasiseinigung bzw. die zu bewertende Regel. Dann werden die Äquivalenzklassen 6 selbst (und nicht die einzelnen Mitglieder dieser Äquivalenzklassen) bewertet, um einen Tupel-Satz 8 von Äquivalenzklassen zu finden, dessen Mitglieder alle den Satz von (ebenfalls mit mehr Einzelheiten später beschriebenen) Constraints erfüllen. Schließlich wird der Satz von Tupel- und von Äquivalenzklassen zu dem Satz von realen Objekten zurückübertragen, um den Satz 9 von Tupeln realer Objekte aus dem Satz realer Objekte 2 zu finden, der den Satz von Constraints erfüllt.

Bei auf Regel beruhenden Expertensystemen, die TREAT-Netze oder RETE-Netze benutzen, werden bezeichnende Constraints identifiziert durch Bestimmen von Attributen von Datenelementen, die mit Attributen anderer Datenelemente verglichen werden. Derartige Constraints können durch eines von zwei Verfahren identifiziert werden. Erstens wird, wenn ein Attribut einer Datenklasse an eine Variable in einer Bedingung einer Regel gebunden ist und diese Variable als eine geprüfte Variable bei einer anderen Bedingung dieser Regel benutzt wird, dieses Attribut dieser Datenklasse als ein bezeichnendes Constraint oder bezeichnendes Attribut angesehen. Zweitens wird, wenn ein Attribut einer Datenklasse gegen eine Variable geprüft wird, die vorher in einer anderen Bedingung dieser Regel gebunden ist, dieses Attribut ebenfalls als bezeichnendes Constraint oder bezeichnendes Attribut angesehen.

In gleicher Weise kann bei Datenbasis-Systemuntersuchungen ein bezeichnendes Constraint identifiziert werden, wenn ein Nicht-Schlüsselgebiet von einer Tabelle mit einem Gebiet von einer anderen Tabelle in einem gemeinsamen Ausdruck verglichen wird.

Fig. 6 ist ein Datenflußdiagramm eines TREAT-Netzes 490, das gemäß der vorliegenden

Erfindung arbeitet. In Fig. 6 ist der TREAT-Knoten in der gleichen Weise wie in Fig. 3 angeordnet. Wenn nicht besonders beschrieben, arbeiten die Wahlknoten 410a - 410d und die Verbindungsknoten 420a - 420d in der gleichen Weise wie die entsprechenden Wahl- bzw. Verbindungsknoten in Fig. 3 und werden nicht nachfolgend im einzelnen beschrieben. Wie in Fig. 3 bewertet der Wahlknoten 410d alle durch den Wahlknoten 410c an seinen Eingangsanschluß zugeführten Zeichen und verarbeitet nur solche Zeichen, welche SOKEN-Klassen-Datenelemente repräsentieren. Alle solche Zeichen sind in seinem zugeordneten internen Speicher 411d gespeichert. Jedoch sind solche Zeichen, die in dem internen Speicher 411d des Wahlknotens 410d gespeichert sind, auch in einer Äquivalenzklassen-Tabelle 412d gespeichert.

Jeder Eintrag in der Äquivalenzklassen-Tabelle 412d besteht aus einer Äquivalenzklassen-Zahl (EC), welche diese Äquivalenzklasse identifiziert; und dieser Äquivalenzklasse zugeordneten Mitgliedern. Die Äquivalenzklassen-Zahl EC kann durch irgendeinen Standard-Zufallsalgorithmus erzeugt werden aufgrund von einem bezeichnenden Attribut, nach welchem der entsprechende Wahlknoten prüft. Mitglieder irgendeiner Äquivalenzklasse bestehen aus einer ID-Nummer (bezeichnet mit einem Vorsatz "#" in der Äquivalenzklassen-Tabelle 412d), welche dieses Teil identifiziert, und Attributen, die dem durch diese ID-Nummer identifizierte Mitglied zugeordnet sind. Die Attribut-Information der Mitglieder der Äquivalenzklassen-Tabelle wird erzeugt aus der entsprechenden Information von dem in jedem durch den Wahlknoten gewählten Zeichen enthaltenen Datenelement.

Im Falle des Wahlknotens 410d ist ^FARBE das bezeichnende Attribut der SOCKEN-Klassen-Datenelemente in Regel 60 (der Fig. 2). Das kann bestimmt werden, da das ^FARBE-Attribut an eine Variable <COL> in der Bedingung der Zeile 70 der Regel 60 gebunden ist und die Variable <COL> in einer anderen Bedingung der Regel 60 (Zeilen 72-76) geprüft wird. Deshalb besteht für jedes eindeutig bewertete ^FARBE-Attribut in den SOCKEN-Klassen-Datenelementen eine eindeutige Äquivalenzklasse, identifiziert durch eine entsprechende Äquivalenzklassen-Zahl EC. Jeder eindeutigen Äquivalenzklasse wird dann ein Satz von Attributen (^FARBE, ^ZUSTAND {sauber oder schmutzig} und ^ALTER) jedes Datenelementes von der Arbeitsdatenbasis zugeordnet, das durch die Zeichen, die die Prüfung des Wahlknotens 410d durchlaufen haben, dargestellt wird.

Wenn das erste Zeichen der SOCKEN-Klasse, das durch den Wahlknoten 410d hindurchtritt, ein ^FARBE-Attribut mit dem Wert schwarz hat (ein Paar schwarze Socken repräsentierend), wird eine Äquivalenzklassen-Zahl, z.B. E-4-1 in der Äquivalenzklassen-Tabelle 412d für alle Datenelemente geschaffen, die schwarze Socken repräsentieren, ohne Rücksicht

auf die Werte der anderen Attribute der SOCKEN-Klassen-Datenelemente. Zusätzlich wird eine ID-Nummer geschaffen, z.B. #234 in dem Ausführungsbeispiel nach Fig. 6, die ein sauberes schwarzes Paar von Socken, das ein Jahr alt ist, mit der Äquivalenzklassen-Zahl E-4-1 verbindet. Da eine neue Äquivalenzklasse geschaffen wurde, wird ein die Äquivalenzklasse E-4-1 repräsentierendes Zeichen zu dem Verbindungsknoten 420d weitergeleitet.

Wenn das nächste durch den Wahlknoten 410d hindurchtretende Zeichen ebenfalls ein Paar schwarze Socken repräsentiert, wird keine neue Äquivalenzklassen-Zahl geschaffen, da eine Schwarzsocken-Äquivalenzklassennummer bereits existiert, d.h. in diesem Beispiel E-4-1. Stattdessen wird eine neue ID-Nummer, d.h. #211 geschaffen, die ein sauberes Paar schwarze Socken, das 2 Jahre alt ist, der Äquivalenzklassen-Nummer E-4-1 zuordnet. Innerhalb einer Äquivalenzklasse können Mitglieder dieser Äquivalenzklasse als eine standardmäßig verbundene Liste gespeichert werden, wie in der Klassentabelle 412d gezeigt (wodurch die Speicherungsanforderungen minimiert werden), oder sie können in einer Zufallszahl-Tabelle aufgrund ihrer ID-Nummern gespeichert werden (wodurch die zum Entfernen von Mitgliedern erforderliche Suchzeit minimiert wird). Da keine neue Äquivalenzklasse geschaffen wurde, wird in diesem Falle kein Zeichen zum Verbindungsknoten 420d durchgeleitet.

Wenn das nächste durch den Wahlknoten 410b hindurchtretende Zeichen ein Paar weiße Socken repräsentiert, wird eine neue Äquivalenzklassen-Nummer E-4-2 für alle Elemente geschaffen, die weiße Socken repräsentieren. In der Äquivalenzklassen-Tabelle 412d wird die EC E-4-2 für Datenelemente geschaffen, die weiße Socken repräsentieren. Dieses Zeichen hat dann eine ihr zugeordnete ID-Nummer, z.B. #69 bei dem vorliegenden Beispiel, und wird ein Eintrag, der mit dem sauberen Paar weiße Socken, die 2 Jahre alt sind, verbunden ist. Ein die neu geschaffene Äquivalenzklasse E-4-2 repräsentierendes Zeichen wird zum Verbindungsknoten 420d durchgeleitet. Wenn das nächste durch den Wahlknoten 410d hindurchtretende Zeichen ein Paar schwarze Socken repräsentiert, braucht keine neue Äquivalenzklassen-Nummer geschaffen zu werden, da E-4-1 bereits für schwarze Socken vorhanden ist. Dieses neue Element wird mit EC E-4-1 assoziiert und bekommt eine ID-Nummer #41. Dieser Eintrag repräsentiert ein sauberes Paar schwarzer Socken, die 3 Jahre alt sind. Kein Zeichen wird zum Verbindungsknoten 420d durchgeleitet. Wenn das nächste durch den Wahlknoten 410d hindurchtretende Zeichen ein Paar blaue Socken repräsentiert, wird eine neue Äquivalenzklassen-Nummer EC E-4-3 für alle Datenelemente geschaffen, die blaue Socken repräsentieren. In der Äquivalenzklassen-Tabelle 412d wird EC E-4-3 für blaue Socken geschaffen. Für dieses Zeichen wird dann eine ID-Nummer geschaffen, z.B. im gegenwärtigen Beispiel #158, und wird mit dem sauberen Paar blaue Socken verbunden, die 2 Jahre alt sind. Ein die

neu geschaffene Äquivalenzklasse E-4-3 repräsentierendes Zeichen wird zum Verbindungsknoten 420d durchgeleitet.

Dieses Verfahren zum Schaffen von neuen Äquivalenzklassen für Datenelemente mit eindeutig bewerteten bezeichnenden Attributen, das jedem der Datenelemente eine ID-Nummer zuordnet und das Datenelement seiner entsprechenden Äquivalenzklasse zuordnet, sowie Zeichen, welche die neu geschaffene Äquivalenzklassen repräsentieren, zu darauffolgenden Verbindungsknoten durchleitet, hält an, bis keine weiteren Zeichen durch den Wahlknoten 410d hindurchtreten.

Es ist auch möglich, in einer bekannten Weise Datenelemente zu beseitigen. Wenn ein Datenelement beseitigt wird, wird ein dieses Datenelement repräsentierendes Zeichen durch das TREAT-Netz 490 mit der gleichen Information, jedoch mit einer Anzeige geleitet, daß dieses Element zu beseitigen ist. Wenn ein solches Zeichen durch den Wahlknoten 410d hindurchgeleitet wird, und es sich auf einen Eintrag in der Äquivalenzklassen-Tabelle 412d bezieht, wird dieser Eintrag aus der Äquivalenzklassen-Tabelle 412d beseitigt.

Wenn beispielsweise durch den Wahlknoten 410d ein Beseitigungszeichen hindurchgeleitet wird, das das Datenelement mit der ID-Nummer #41 in der Äquivalenzklasse E-4-1 repräsentiert, wird dieses Datenelement aus der Äquivalenzklassen-Tabelle 412d beseitigt. Wenn durch den Wahlknoten 410d ein Beseitigungszeichen durchgeleitet wird, das das Datenelement mit der ID-Nummer #69 in Äquivalenzklasse E-4-2 entspricht, wird nicht nur dieses Datenelement aus der Äquivalenzklassen-Tabelle, sondern die Äquivalenzklasse selbst beseitigt. Zusätzlich werden die Einträge in dem Konfliktsatz im Ergebnisknoten 430, die sich auf diesen Äquivalenzsatz beziehen, ebenfalls in einer bekannten Weise beseitigt. Das Beseitigen von Datenelementen wird später besprochen.

Die Wahlknoten 410a - 410c arbeiten in einer gleichartigen Weise, um Datenelemente, die ihre jeweilige Prüfung bestanden haben, in ihren zugeordneten Speichern 411a - 411c als Äquivalenzklassen-Tabellen zu speichern. Dann werden statt des Durchleitens von Zeichen, welche passende Datenelemente darstellen, zu den jeweiligen Verbindungsknoten, 420a - 420d nur Zeichen hindurchgeleitet, welche passende Äquivalenzklassen-Nummern ECs repräsentieren.

Die Verbindungsknoten 420a - 420d wirken in einer gleichartigen Weise wie Verbindungsknoten 120a - 120d in Fig. 3, doch leiten sie, statt Datenpakete zur Bewertung von Attributen von einzelnen Datenelementen, die durch die zu ihnen durchgeleiteten Zeichen repräsentiert werden (wie in Fig. 3), Datenpakete zum Bewerten der bezeichnenden Attribute der Äquivalenzklassen durch, die durch die zu ihnen durchgeleiteten Zeichen repräsentiert werden. Zei-

chen, welche Kombinationen von Äquivalenzklassen mit bezeichnend n Attributen repräsentieren, die die dem TREAT-Netz entsprechenden Prüfungen erfüllen, werden dann weiter zu dem Ergebnisknoten 430 geleitet, um Mitglieder des Konfliktsatzes 431 zu werden. Der im Ergebnisknoten 430 enthaltene Konfliktsatz 431 enthält dann einen Satz von Zeichen, welche Kombinationen von Äquivalenzklassen mit bezeichnenden Attributen repräsentieren, deren Werte die dem TREAT-Netz entsprechenden Regel erfüllen, und nicht Kombinationen von einzelnen Datenelementen mit Attributen, deren Werte die Regel erfüllen. Weitere Verarbeitung der Zeichen in dem Konfliktsatz 431 wird im einzelnen später beschrieben.

Fig. 7 ist ein Datenfluß-Schaubild eines RETE-Netzes 390 entsprechend der Regel 60 der Fig. 2 gemäß der vorliegenden Erfindung. In Fig. 7 arbeiten, wenn es nachstehend nicht anders beschrieben ist, Elemente, die denen in Fig. 5 entsprechen, in einer gleichartigen Weise, und werden im einzelnen nachfolgend nicht beschrieben. In Fig. 7 sind alle durch den 1-Eingangs-Knoten 310d empfangenen Zeichen in dessen zugeordnetem Speicher 311d gespeichert. jedoch sind die in dem Speicher 311d des 1-Eingangs-Knotens 310d gespeicherten Datenelemente in einer Äquivalenzklassen-Tabelle 312d gespeichert. Jeder Eintrag in der Äquivalenzklassen-Tabelle 312d besteht aus einer Äquivalenzklassen-Nummer EC und Mitgliedern, die dieser Äquivalenzklassen-Nummer zugeordnet sind. Mitglieder der Äquivalenzklassen-Tabelle bestehen aus einer ID-Nummer und Attributen (^FARBE, ^ZUSTAND {sauber oder schmutzig} und ^ALTER) des Datenelementes, das dieser ID-Nummer entspricht. Die Äquivalenzklassen-Nummer kann durch irgendeinen Standard-Zufallsalgorithmus erzeugt werden mit Hilfe des bezeichnenden Attributs, aufgrund dessen der entsprechende 1-Eingangs-Knoten prüft.

Im Falle des 1-Eingangs-Knotens 310d ist das bezeichnende (auf die vorher beschriebene Weise bestimmte) Attribut der SOCKEN-Klassen-Datenelemente das Attribut ^FARBE. Deshalb besteht für jedes Datenelement der SOCKEN-Klasse mit eindeutig bewertetem ^FARBE-Attribut (das ein eindeutig gefärbtes Sockenpaar repräsentiert) eine eindeutige Äquivalenzklasse, die durch eine EC-Nummer identifiziert ist. Zu jeder eindeutigen Äquivalenzklasse sind dann die Attribute aktueller Datenelemente zugeordnet, die in Zeichen enthalten sind, welche die Prüfung des entsprechenden 1-Eingangs-Knotens 310d bestanden haben.

In Fortsetzung des Socken-Beispiels wird, wenn das erste durch den 1-Eingangs-Knoten 310d hindurchtretende Zeichen ein Datenelement der SOCKEN-Klasse repräsentiert, mit einem ^FARBE-Attribut mit dem Wert schwarz (ein Paar schwarzer Socken repräsentierend), eine Äquivalenzklassen-Nummer EC, z.B. E-4-1 in der Äquivalenzklassen-Tabelle 312d, für Datenelemente erzeugt, welche schwarze Socken repräsentieren. Zusätzlich dazu wird eine

ID-Nummer geschaffen, z.B. in Fig. 7 #234, und die Attribute des gegenwärtigen Zeichens werden in diesen Eintrag in der Äquivalenzklassen-Tabelle kopiert. Das ordnet einem sauberen Paar schwarze Socken, das 1 Jahr alt ist, die Äquivalenzklassen-Nummer E-4-1 zu. Da eine neue Äquivalenzklasse geschaffen wurde, wird ein die neu geschaffene Äquivalenzklasse E-4-1 repräsentierendes Zeichen, das den Wert des bezeichnenden Attributes dieser Äquivalenzklasse enthält, zu dem 2-Eingangs-Knoten 320c durchgeleitet. (Die anderen nicht bezeichnenden Attribute sind in dem die Äquivalenzklasse repräsentierenden Zeichen nicht eingeschlossen. Wenn diese Attribute für ein Datenelement benötigt werden, das ein Mitglied dieser Äquivalenzklasse ist, können sie von dem zugehörigen, dieses Datenelement repräsentierenden Äquivalenzklassen-Tabelleneintrag abgeleitet werden.)

Wenn das nächste durch den 1-Eingangs-Knoten 310d hindurchtretende Zeichen ein weiteres Paar schwarze Socken repräsentiert, wird keine neue Äquivalenzklassen-Nummer geschaffen, da bereits eine Äquivalenzklassen-Nummer für schwarze Socken vorhanden ist, d.h. in diesem Beispiel E-4-1. Eine neue ID-Nummer, d.h. #211, wird für dieses Zeichen geschaffen, und die Attribute des in dem Zeichen enthaltenen Datenelements werden in diesen Eintrag in die Äquivalenzklassen-Tabelle kopiert. Das ordnet einem sauberen Paar schwarze Soken, das 2 Jahre alt ist, die Äquivalenzklassen-Nummer E-4-1 zu. Innerhalb einer Äquivalenzklasse können Mitglieder dieser Äquivalenzklasse als eine Standard-Verbindungsliste gespeichert werden, wie in der Klassentabelle 412d gezeigt (wodurch die Speicherungsanforderungen minimiert werden), oder sie können in einer Zufallstabelle aufgrund ihrer ID-Nummern gespeichert werden (wodurch die zum Entfernen von Mitgliedern erforderliche Suchzeit minimiert wird). In diesem Fall wird kein Zeichen zu dem 2-Eingangs-Knoten 320c durchgeleitet.

Wenn das nächste durch den 1-Eingangs-Knoten 310d hindurchtretende Zeichen ein Paar weiße Socken repräsentiert, wird eine neue Äquivalenzklassen-Nummer E-4-2 für alle Datenelemente geschaffen, die weiße Socken repräsentieren. In der Äquivalenzklassen-Tabelle 312d wird die Äquivalenzklassen-Nummer EC E-4-2 für Datenelemente geschaffen, die weiße Socken repräsentieren. Der Äquivalenzklassen-Tabelleneintrag, der das erste Paar weiße Socken repräsentiert, besitzt dann eine geschaffene ID-Nummer, z.B. #69 bei dem gegenwärtigen Beispiel, und dieser Eintrag wird dem sauberen Paar weiße Socken zugeordnet, die 2 Jahre alt sind, repräsentiert durch das gegenwärtige Zeichen. Ein Zeichen, das die neu geschaffene Äquivalenzklasse E-4-2 repräsentiert, wird zum 2-Eingangs-Knoten 320c durchgeleitet. Wenn das nächste durch den 1-Eingangs-Knoten 310d hindurchtretende Zeichen ein Paar schwarze Socken repräsentiert, braucht keine neue Äquivalenzklassen-Nummer ge-

schaffen zu werden, da E-4-1 bereits für schwarze Socken vorhanden ist. Dieses neue Datenelement wird E-4-1 zugeordnet und besitzt eine ID-Nummer #41. Das repräsentiert ein sauberes Paar schwarze Socken, die 3 Jahre alt sind. Wiederum wird kein Zeichen zum 2-Eingangs-Knoten 320c durchgeleitet.

Wenn das nächste durch den 1-Eingangs-Knoten 310d hindurchtretende Zeichen ein Paar blaue Socken repräsentiert, wird eine neue Äquivalenzklassen-Nummer E-4-3 für alle Datenelemente geschaffen, die blaue Socken repräsentieren. In der Äquivalenzklassen-Tabelle 312d wird der Äquivalenzklassen-Nummer EC E-4-3 für Datenelemente geschaffen, die blaue Socken repräsentieren. Das erste, ein Paar blaue Socken repräsentierende Datenelement hat dann eine geschaffene ID-Nummer, d.h. #158 bei dem gegenwärtigen Beispiel und die Attribute dieses Datenelements werden in dem Eintrag gespeichert und der Eintrag wird dem sauberen Paar blaue Socken, die 2 Jahre sind, zugeordnet. Ein die neu geschaffene Äquivalenzklasse E-4-3 repräsentierendes Zeichen wird zum 2-Eingangs-Knoten 320c hindurchgeleitet.

Dieser Vorgang des Schaffens neuer Äquivalenzklassen für Datenelemente mit eindeutig bewerteten bezeichnenden Attributen unter Zuordnung jedes der Datenelemente über eine ID-Nummer zu ihrer entsprechenden Äquivalenzklasse und des Durchleitens von neu geschaffenen Äquivalenzklassen repräsentierenden Zeichen zu nachfolgenden 2-Eingangs-Knoten hält an, bis keine weiteren Zeichen durch den 1-Eingangs-Knoten 310d hindurchtreten. Die Beseitigung von Datenelementen von den Äquivalenzklassen-Tabellen im RETE-Netz 390 wird in der gleichen Weise behandelt, wie in dem TREAT-Netz 490 (der Fig. 6), bis zu dem Punkt, daß Beseitigungszeichen, welche die Äquivalenzklassen repräsentieren, die beseitigt wurden, zu dem darauffolgenden 2-Eingangs-Knoten durchgeleitet wurden.

Die 1-Eingangs-Knoten 310a - 310c arbeiten in einer gleichartigen Weise zum Speichern von Datenelementen, die die jeweiligen Prüfungen durch den 1-Eingangs-Knoten bestehen, in den internen Speicher 311a - 311c als Einträge in jeweilige Äquivalenzklassen-Tabellen. Dann werden, statt daß einzelne Zeichen zu jeweiligen 2-Eingangs-Knoten 320a - 320c weitergeleitet werden, welche passende entsprechende Datenelemente repräsentieren, nur Zeichen weitergeleitet, die neu geschaffene Äquivalenzklassen-Nummern EC repräsentieren.

Im allgemeinen werden Zeichen, die Äquivalenzklassen oder Sätze von Äquivalenzklassen repräsentieren, an den linken Eingangsanschlüssen von 2-Eingangs-Knoten empfangen, und Zeichen, die einzelne Äquivalenzklassen repräsentieren, an den rechten Eingangsanschlüssen der 2-Eingangs-Knoten empfangen, in der gleichen Weise wi Zeichen, die Datenelemente oder Sätze von Datenelementen in den in Fig. 4 dargestellten 2-Eingangs-Knoten nach dem

Stand der Technik empfangen werden. Diese Zeichen werden auch durch die 2-Eingangs-Knoten in einer gleichartigen Weise verarbeitet: Vergleichen der bezeichnenden Attribute der durch die neuerdings empfangenen Zeichen repräsentierten Äquivalenzklassen an einem Eingangsanschluß für die bezeichnenden Attribute der Äquivalenzklassen, die durch die jeweiligen, vorher in den mit dem anderen Eingangsanschluß verbundenen Speicher gespeicherten Zeichen repräsentiert werden. Wenn die bezeichnenden Attribute die durch den 2-Eingangs-Knoten durchgeführten Untersuchungen bestehen, wird ein Zeichen erzeugt, das die Kombination der Äquivalenzklassen in den beiden passenden Zeichen repräsentiert.

Als ein Beispiel werden Zeichen, welche die Äquivalenzklassen-Nummern E-4-1, E-4-2, E-4-3 von der Äquivalenzklassen-Tabelle 312d im internen Speicher 311d des 1-Eingangs-Knotens 310d repräsentieren, dem rechten Eingangsanschluß des 2-Eingangs-Knotens 320c zugeleitet und in dem rechten Speicher 321f in gleichartiger Weise gespeichert wie Zeichen, die Datenelemente repräsentieren, in dem RETE-Netz 290 der Fig. 4 gespeichert werden. Wenn sie empfangen werden, werden diese Zeichen mit allen Zeichen verglichen, welche die Kombinationen von Äquivalenzklassen repräsentieren, die vorher von dem 2-Eingangs-Knoten 320b erhalten und in dem linken Speicher 321c gespeichert wurden. Wenn Paare von Zeichen gefunden werden, welche die Prüfung des 2-Eingangs-Knotens 220c bestehen, werden Zeichen, welche die Kombination von Äquivalenzklassen der beiden angepaßten Zeichen repräsentieren, erzeugt und zu dem Ergebnisknoten 330 durchgeleitet, um in den Konfliktsatz eingesetzt zu werden. Der Konfliktsatz im Ergebnisknoten 330 enthält dann einen Satz von Zeichen, welcher Kombinationen von Äquivalenzklassen repräsentiert, dessen bezeichnende Attribute die dem RETE-Netz 390 entsprechende Regel erfüllen und nicht Kombinationen von einzelnen Datenelementen, deren Attribute die Regel erfüllen.

Die weitere Verarbeitung der Konfliktsätze 431 in dem TREAT-Netz 490 nach Fig. 6 und der Konfliktsätze 332 in dem RETE-Netz 390 nach Fig. 7 werden nachfolgend beschrieben. Um die Erklärung zu vereinfachen, wird nachfolgend nur der Konfliktsatz 332 in dem RETE-Netz 390 der Fig. 7 besprochen, aber diese Besprechung gilt auch für die Verarbeitung beider Konfliktsätze, falls nicht anders festgestellt wird.

Jedes zu dem Ergebnisknoten 330 durchgeleitete Zeichen stellt nicht einen einzelnen Satz von Datenelementen dar, der die Bedingungen der Regel erfüllt, sondern repräsentiert stattdessen einen Satz von Äquivalenzklassen, deren bezeichnende Attribute die Bedingungen der Regel erfüllen. Wie vorstehend beschrieben, kann jede Äquivalenzklasse eine Vielzahl von aktuellen Datenelementen aufweisen, die zu ihr gehören.

Wenn ein Zeichen im Ergebnisknoten 330 empfangen wird, um in den Konfliktsatz 332

eing setzt zu werden, werden die in dem diesem Zeichen identifizierten Äquivalenzklassen (z.B. E-1-1, E-2-3, E-3-5 und E-4-2) entsprechend gut bekannten Verfahren bewertet, um die "beste" Kombination von Datenelementen zu wählen, die zu den jeweiligen Äquivalenzklassen gehören, die nicht vorher zum Feuern der Regel benutzt worden sind. Das kann dadurch erledigt werden, daß die jeweilige Äquivalenzklassen-Nummern EC, die in den ausgewählten Zeichen in dem Konfliktsatz enthalten sind, zum Zugriff zu den jeweiligen Äquivalenzklassen-Tabellen benutzt werden, welche die aktuellen Datenelemente aufweisen, die Mitglieder dieser Äquivalenzklassen sind. (Das wird im einzelnen später besprochen.) Sowohl die Äquivalenzklassen-Identifizierer als auch die beste unbenutzte Kombination von Datenelementen, die zu den jeweiligen Äquivalenzklassen gehören, sind in jedem in dem Konfliktsatz gespeicherten Zeichen gespeichert.

Die Datenelement-Kombinationen in jedem Zeichen in dem Konfliktsatz 332 werden ebenfalls in Übereinstimmung mit gut bekannten Verfahren bewertet, um die "beste" solcher Kombinationen zu wählen, die dann als die angesehen wird, die das Feuern der Regel verursacht. Nachdem eine Regel zum Feuern gewählt worden ist, jedoch bevor sie tatsächlich gefeuert wird, wird die nächstbeste unbenutzte Kombination von Datenelementen von den in dem gewählten Zeichen von dem Konfliktsatz 332 enthaltenen jeweiligen Äquivalenzklassen bestimmt. Wenn eine solche Kombination vorhanden ist, ersetzen solche Datenelemente die Kombination von Datenelementen in dem Konfliktsatz-Zeichen, die als die angesehen werden, die die Regel zum Feuern gebracht haben.

Wenn keine solche Kombination vorhanden ist, wird dieses Konfliktsatz-Zeichen als inaktiv markiert. Wenn ein neues Datenelement in irgendeiner der Äquivalenzklassen-Tabellen erhalten wird, auf die sich dieses Konfliktsatz-Zeichen bezieht, dann wird dieses Konfliktsatz-Zeichen wieder aktiv gemacht und der genannte Vorgang wiederholt.

Um die beste unbenutzte Kombination von Datenelementen aufrecht zu erhalten, die mit jedem Konfliktsatz-Zeichen verbunden ist, hält jede Äquivalenzklasse in den jeweiligen Äquivalenzklassen-Tabellen eine Liste der Konfliktsatz-Zeichen aufrecht, welche Bezüge zu dieser Äquivalenzklasse enthalten. Jedesmal, wenn ein neues Datenelement einer Äquivalenzklasse hinzugefügt wird, wird, da es neu ist und deshalb (gemäß dem vorher erwähnten bekannten Auswahlverfahren) besser sein kann, die beste unbenutzte Kombination von Datenelementen, die jedem Konfliktsatz-Zeichen zugeordnet ist, das diese Äquivalenzklasse enthält, neu bewertet, um zu bestimmen, ob sie zum Enthalten dieses neu n Datenel ments zu aktualisieren ist. In gleicher Weise muß, wenn ein Mitglied einer Äquivalenzklasse entfernt wird, jedes Konfliktsatz-Zeichen, das diese Äquivalenzklasse enthält, überprüft werden. Wenn die beste unbe-

nutzte Kombination sich nicht auf das beseitigte Datenelement bezieht, dann braucht nichts unternommen zu werden. Wenn di beste unb nutzte Kombination das entfernte Datenelement enthält, muß die beste unbenutzte Kombination neubewertet werden, um zu bestimmen, ob sie aktualisiert werden sollte, damit sie dieses neue Datenelement enthält.

In dem RETE-Netz nach dem Stand der Technik gemäß Fig. 4 wird, nachdem eine Regel gefeuert hat, das Zeichen in dem Konfliktsatz, das die Kombination von Datenelementen enthält, von denen angenommen wird, daß sie die Regel zu feuern veranlaßt haben, von dem Konfliktsatz entfernt werden. Das war ausreichend, um zu bezeichnen, daß diese Kombination von Datenelementen benutzt worden und nicht länger zum Feuern der Regel auswählbar ist. Im RETE-Netz 390 gemäß der vorliegenden Erfindung kann jedoch, nachdem die Regel gefeuert hat, ein Zeichen in dem Konfliktsatz, von dem angenommen wird, daß es das Feuern der Regel verursacht hat, nicht beseitigt werden, weil sich immer noch unbenutzte Datenelemente in den Äquivalenzklassen-Tabellen befinden können. Jedoch müssen die Datenelemente, von denen vorher angenommen wurde, daß sie das Feuern der Regel verursacht haben, für ein neuerliches Feuern der Regel unwählbar gemacht werden.

Das wird dadurch erledigt, daß ein Brechungssatz 335 aufrechterhalten wird. Der Brechungssatz 335 ist eine Liste, welche besonderen Datenelemente vorher eingeschätzt wurden, die Regel gefeuert zu haben. Nachdem ein Zeichen in dem Konfliktsatz 332 gewählt wurde, wird ein Eintrag zu dem Brechungssatz 335 hinzugefügt, der die Kombination von Datenelementen bezeichnet, die zum Feuern dieser Regel benutzt wurden.

Der Auswahlvorgang, um die beste unbenutzte Kombination von Datenelementen entsprechend einem Konfliktsatz-Zeichen zu bestimmen, enthält den Zugriff zu allen Äquivalenzklassen-Tabellen, die mit diesem Konfliktsatz-Zeichen verbunden sind. Dann wird die beste Kombination von Datenelementen gemäß dem erwähnten bekannten Wählverfahren gewählt. Der Brechungssatz 335 wird dann geprüft, um zu bestimmen, ob diese Kombination vorher zum Feuern dieser Regel benutzt wurde. Wenn diese Kombination von Datenelementen in dem Brechungssatz vorhanden ist, werden die Äquivalenzklassen-Tabellen überprüft, um die nächstbeste Kombination von Datenelementen zu wählen. Das wird wiederum gegen die Einträge in dem Brechungssatz 335 geprüft. Das wird wiederholt, bis eine unbenutzte Kombination von Datenelementen gefunden und in das entsprechende Konfliktsatz-Zeichen gesetzt wird, oder keine solche Kombinationen von Datenelementen gefunden werden, in welchem Falle das Konfliktsatz-Zeichen als inaktiv markiert wird, wie vorher angeführt. Dieses Verfahren kann ausgeführt werden, um jede Kombination von Datenelementen in den jeweiligen Äquivalenzklassen zu bewerten, oder kann auf Schritterhöhungsbasis ausgeführt werden,

um auf die nächstbeste Kombination von Datenelementen hin zu überprüfen.

Es wurde hier eine bevorzugte Ausführung gezeigt und beschrieben, es ist dabei jedoch zu verstehen, daß verschiedene andere Anpassungen und Abwandlungen hergestellt werden können.

EP.-Az.: 92 906 036.6-2201

DIGITAL EQUIPMENT CORPORATION

28.09.99

PATENTANSPRÜCHE:

1. Experten-System, das ein Computersystem und ein einen Satz von Regeln und entsprechend den Regeln auszuführenden Vorgängen definierendes Quellenprogramm enthält, wobei das Quellenprogramm mit Benutzung eines Kompilierers verarbeitet wird, um ein ausführbares Programm und zugehörige Datenstruktur zu erzeugen;

und das Computersystem einen Speicher besitzt, in dem das ausführbare Programm und die zugehörige Datenstruktur eingeladen sind, wobei der Speicher auch ausgelegt ist, eine Arbeits-Datenbasis von aktuellen, durch das Expertensystem zu verarbeitenden Elementen zu speichern; das Computersystem betreibbar ist, die Datenelemente durch die Datenstruktur zu verarbeiten, um zu bewerten, wenn die Bedingungen einer Regel durch die Datenelemente erfüllt sind, und um diese Regel zu betätigen und dadurch den mit der betätigten Regel verbundenen Vorgang durchzuführen;

dadurch gekennzeichnet, daß das Computersystem mit Benutzung von Tupeln betreibbar ist, um in der Datenstruktur mindestens eine Äquivalenzklasse zu erzeugen, welche einen durch ein gemeinsames Constraint (Begrenzungsrelation) bestimmten Satz von Datenelementen umfaßt, und ein einzelnes Zeichen, das einen Satz von Äquivalenzklassen darstellt, wobei der Satz Äquivalenzklassen umfaßt mit Constraints, welche die Regelbedingungen erfüllen, das Computersystem Mittel besitzt, um von jedem Satz von Äquivalenzklassen die Datenelemente zurückzutragen, welche die Bedingungen der Regeln erfüllen, und dadurch das Ausführen des zugehörigen Vorgangs zu ermöglichen.

2. Expertensystem nach Anspruch 1, **dadurch gekennzeichnet, daß** die Datenstruktur einen Speicher zum Speichern von Datenelementzeichen enthält, wobei ein Datenelementzeichen durch die Datenstruktur in Reaktion auf ein eingegebenes Datenelement erzeugt wird, die Datenstruktur betätigbar ist, die Datenelementzeichen in dem Speicher in Äquivalenzklassen zu speichern, wobei jede Klasse durch das Constraint bestimmt ist und jedes Datenelementzeichen durch das Constraint und durch andere mit dem Datenelement verbundene Attribute identifiziert wird, um ein Äquivalenzklassenzeichen zur Verarbeitung durch die Datenstruktur jedesmal zu erzeugen, wenn eine neue Äquivalenzklasse erzeugt wird, wobei eine neue Äquivalenzklasse jedesmal erzeugt wird, wenn ein Datenelementzeichen in Reaktion auf ein eingegebenes Datenelement mit einen eindeutig bewerteten Constraint erzeugt wird, und die erzeugten Äquivalenzklassenzeichen zu bewerten zum Einrichten dieser Äquivalenzklassen mit Constraints, welche die Bedingungen der Regel er-

füllen, und dadurch einen Konfliktsatz von Einzelzeichen zu erzeugen, wobei jedes Einzelzeichen in dem Konfliktsatz Kombinationen von Äquivalenzklassen darstellt, welch die Bedingungen der Regel erfüllende Constraints besitzen.

3. Expertensystem nach Anspruch 1 oder 2, **dadurch gekennzeichnet, daß** die Constraints Attribute eines Datenelements sind, die mit den Attributen eines anderen Datenelements verglichen werden.

4. Expertensystem nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet, daß** die Datenstruktur ein RETE-Netz ist.

5. Expertensystem nach einem der Ansprüche 1 bis 3, **dadurch gekennzeichnet, daß** die Datenstruktur ein TREAT-Netz ist.

6. Expertensystem nach einem der Ansprüche 2 bis 5, **dadurch gekennzeichnet, daß** das Computersystem betätigbar ist zum:

Bewerten der in jedem Konfliktsatzzeichen identifizierten Äquivalenzklassen, um die beste Kombination von Datenelementen in jeder Äquivalenzklasse zum Betätigen der Regel zu wählen,

Bestimmen in dem die beste Kombination von Datenelementen enthaltenden Konfliktsatzzeichen die nächsten Kombination von Datenelementen, bevor die Regel betätigt wird, und Ersetzen in dem Konfliktsatzzeichen die beste Kombination durch die nächstbeste Kombination, wobei das Konfliktsatzzeichen als inaktiv markiert wird, wenn keine solche Kombination existiert; und Veranlassen der zum Betätigen der Regel ausgewählten Datenelemente, ungeeignet zu sein, die Regel wieder zu zünden.

7. Expertensystem nach Anspruch 6, **dadurch gekennzeichnet, daß** das Computersystem weiter zur Neubewertung jedes Konfliktsatzzeichens betätigbar ist, um die nächstbeste Kombination zu aktualisieren, wenn ein neues Datenelement zu einer Äquivalenzklasse hinzugefügt wird.

8. Expertensystem nach Anspruch 6 oder 7, **dadurch gekennzeichnet, daß** die Datenstruktur einen Speicher enthält, der Zeichen hält, die vorher zur Betätigung einer Regel benutzte Datenelemente darstellt, wobei die Inhalte des Speichers überprüft werden, bevor eine Regel betätigt wird, um sicherzustellen, daß keine ungeeigneten Datenelemente zur Betätigung jener Regel benutzt werden.

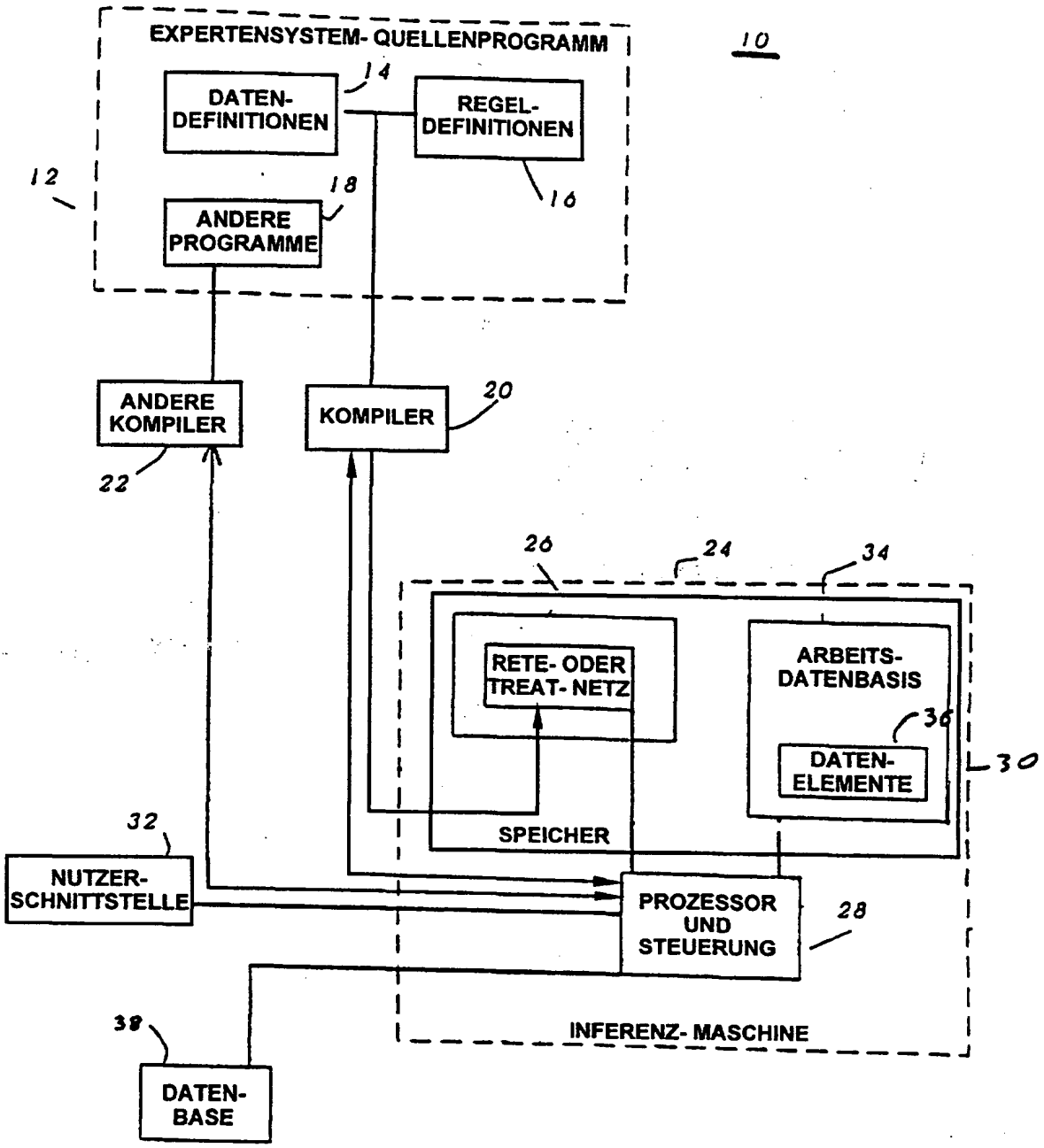

Fig. 1

STAND DER TECHNIK

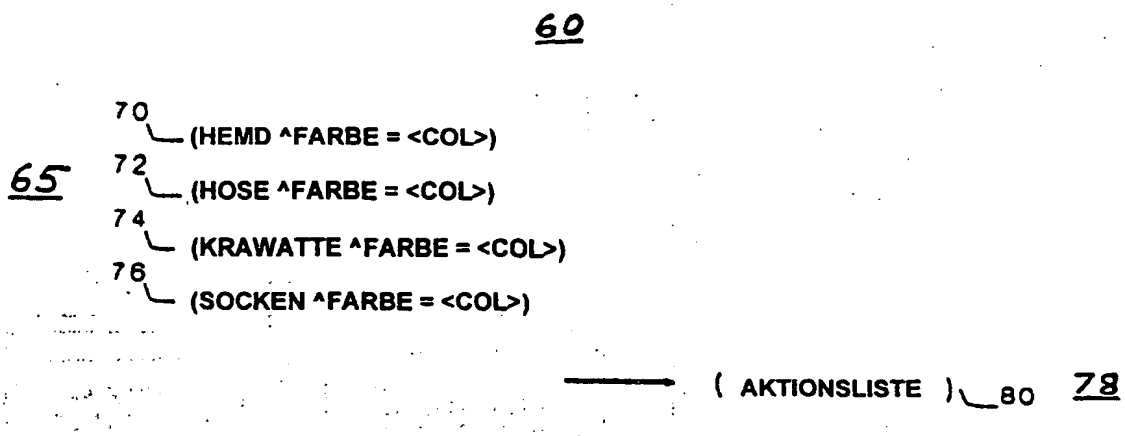

Fig. 2

STAND DER TECHNIK

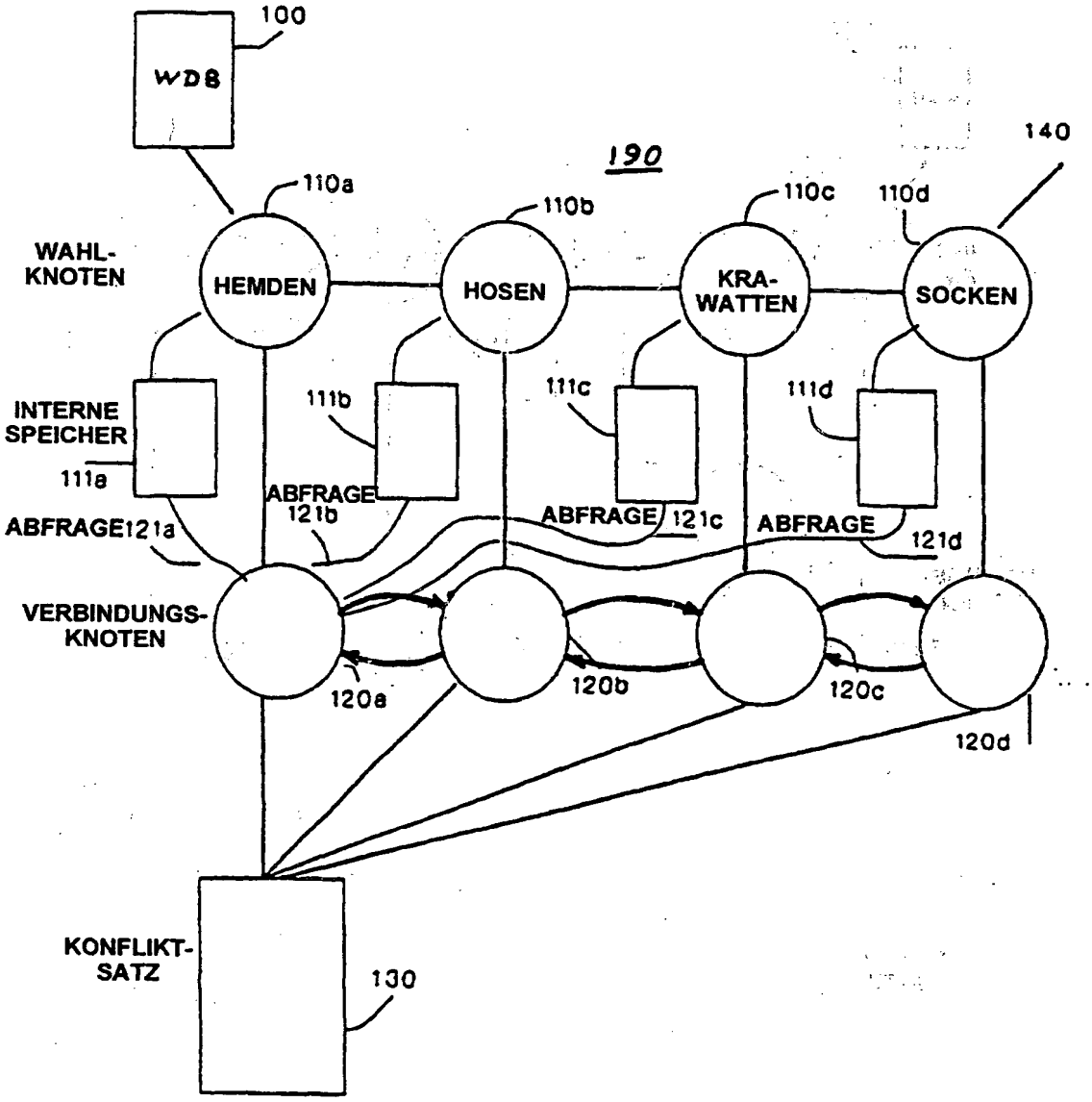

Fig. 3

STAND DER TECHNIK

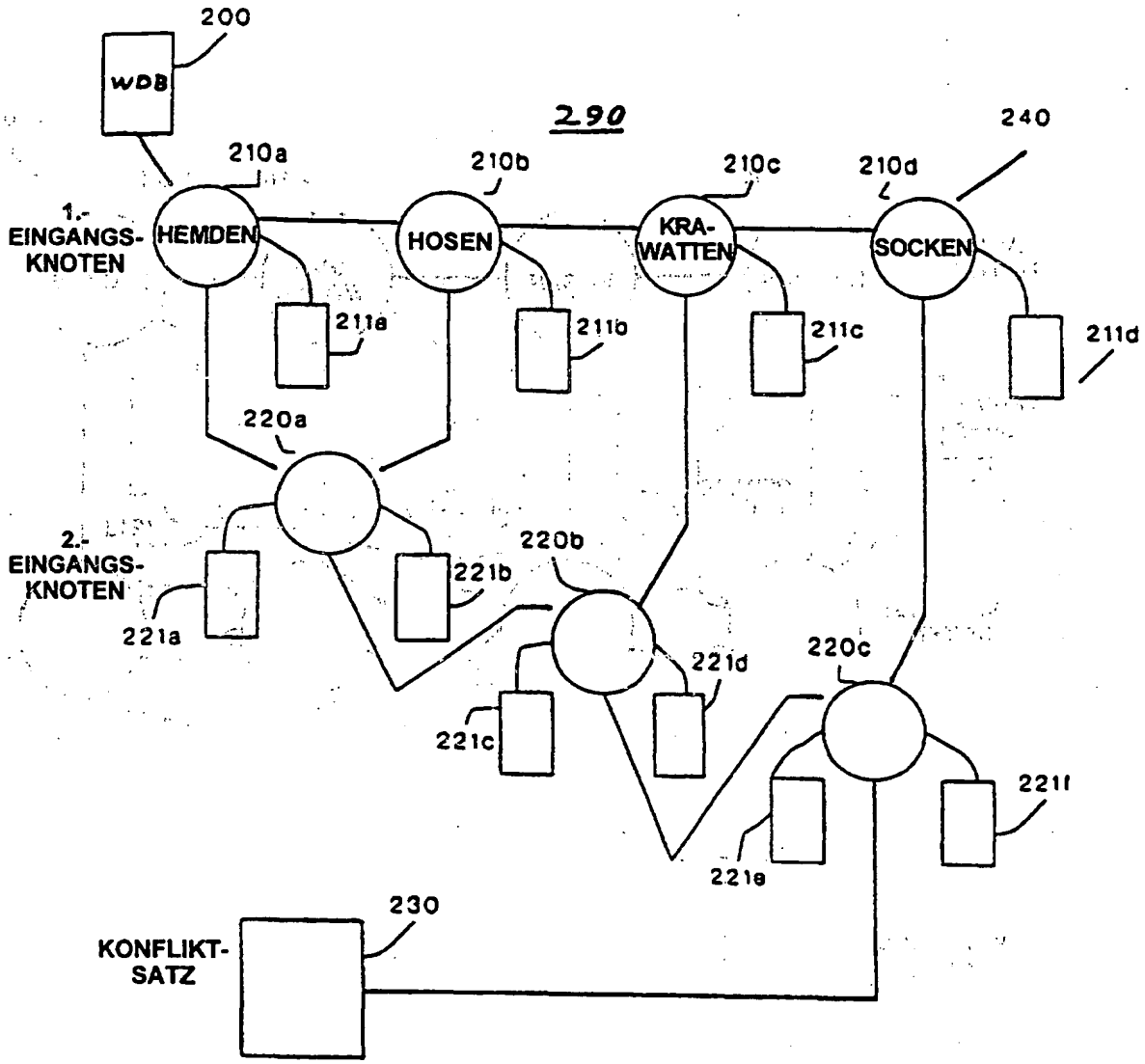

Fig. 4

STAND DER TECHNIK

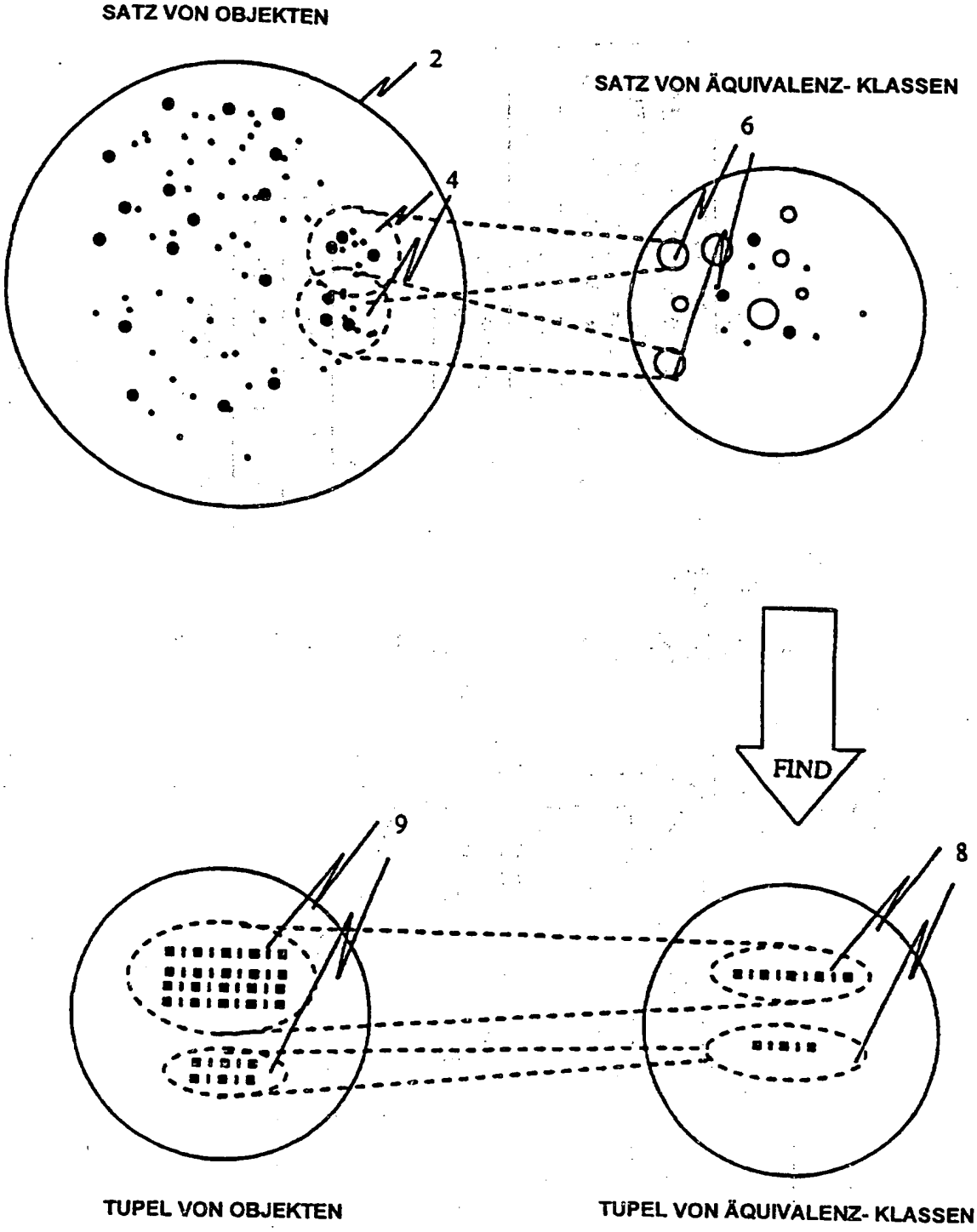
SATZ VON OBJEKTEN
2
SATZ VON ÄQUIVALENZ- KLASSEN
6
4
FIND
9
8
TUPEL VON OBJEKTEN
TUPEL VON ÄQUIVALENZ- KLASSEN

FIG. 5

Herrn Prof. Dr. Augele

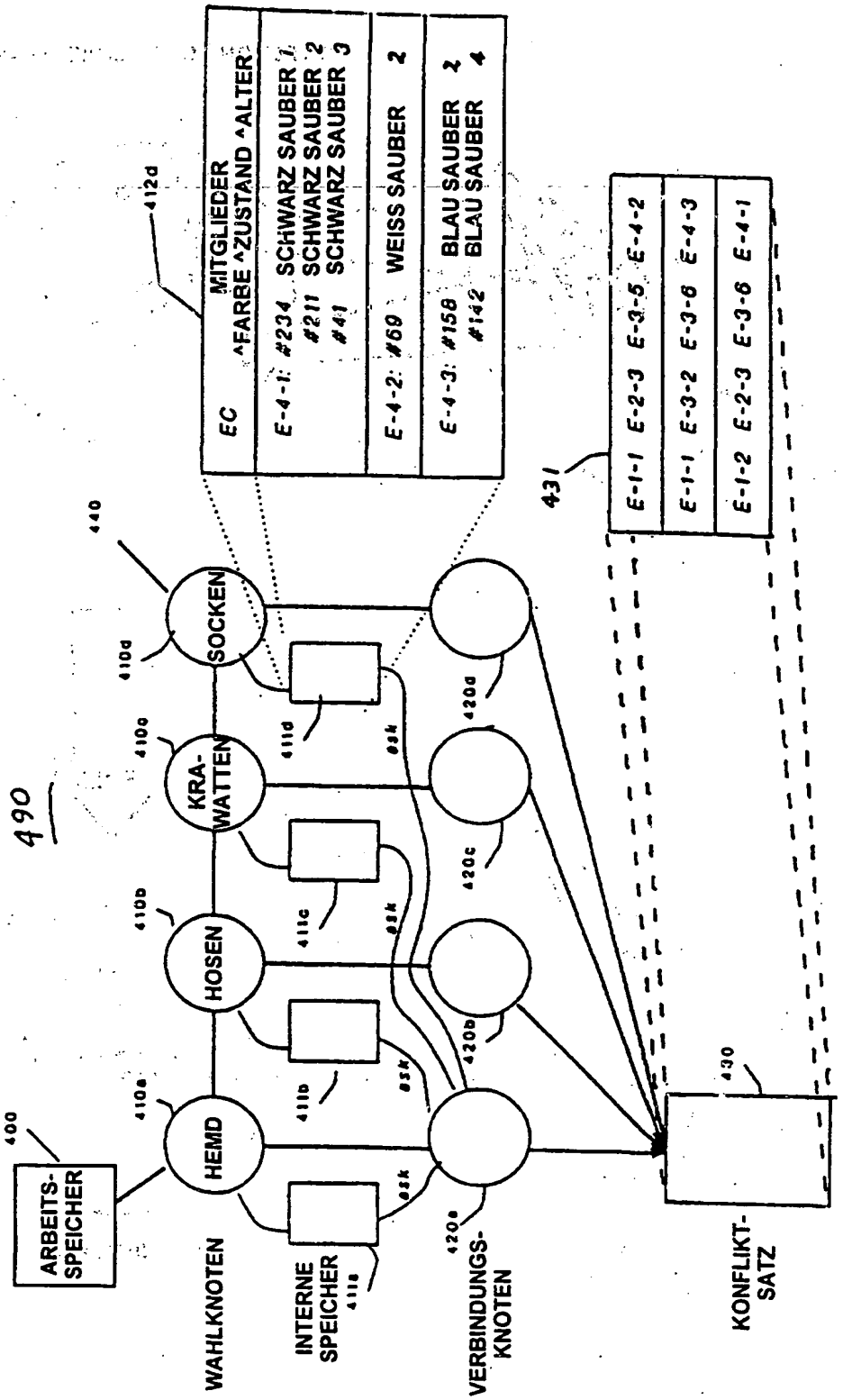

Fig. 6

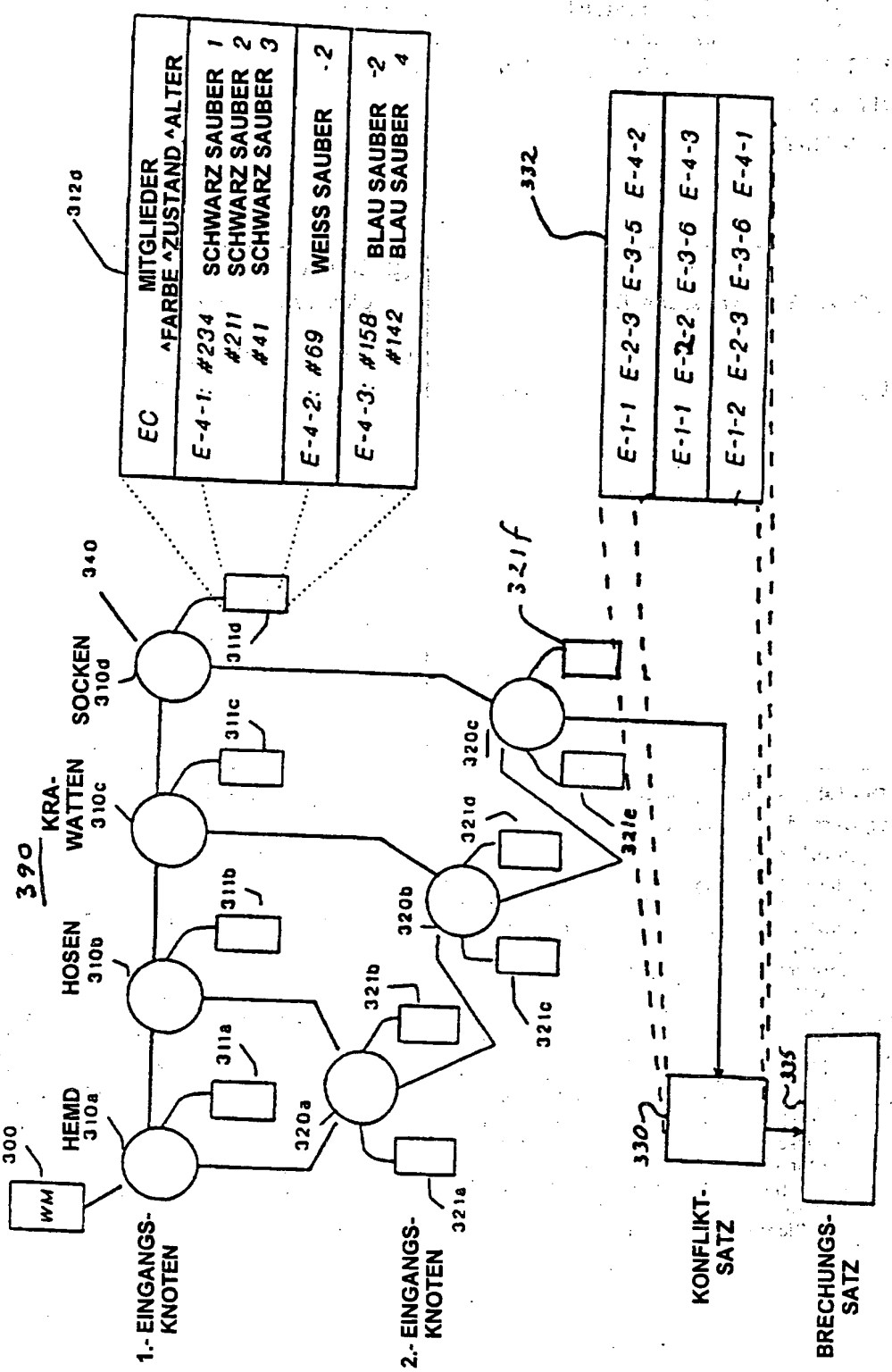
MITGLIEDER
EC ^FARBE ^ZUSTAND ^ALTER
E-4-1: #234 SCHWARZ SAUBER 1
#211 SCHWARZ SAUBER 2
#41 SCHWARZ SAUBER 3
E-4-2: #69 WEISS SAUBER -2
E-4-3: #158 BLAU SAUBER -2
#142 BLAU SAUBER 4
312a
332
E-1-1 E-2-3 E-3-5 E-4-2
E-1-1 E-2-2 E-3-6 E-4-3
E-1-2 E-2-3 E-3-6 E-4-1
340
SOCKEN 310d
KRA-WATTEN 310c
HOSEN 310b
HEMD 310a
390
300
WM
311a
311b
311c
311d
320a
320b
320c
321a
321b
321c
321d
321e
321f
330
335
1.- EINGANGS-KNOTEN
2.- EINGANGS-KNOTEN
KONFLIKT-SATZ
BRECHUNGS-SATZ

FIG. 7